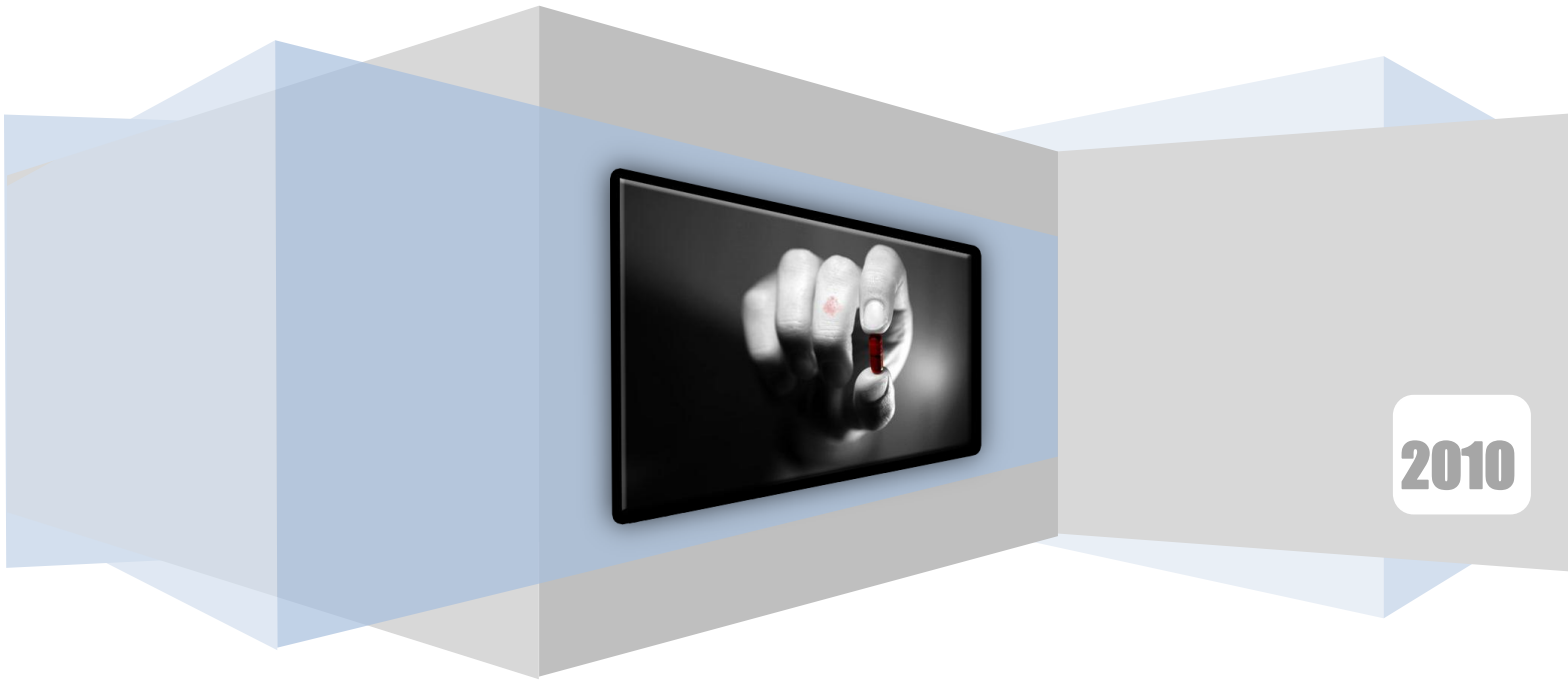
Universidad Carlos III de Madrid

Ingeniería Técnica en Informática de Gestión

SVMDDI:

Un sistema para la extracción de interacciones farmacológicas basado en Support Vector Machines (SVM)

Autor: Víctor Manuel Méndez González
Tutora: Isabel Segura Bedmar



2010

Agradecimientos:

A Isa por saberme motivar en los momentos que más lo necesitaba y por ser tan buena gente, así da gusto hacer el proyecto.

A mi familia, porque estoy aquí gracias a vuestro esfuerzo continuo y porque me siento feliz al formar parte de ella.

A Kathy por transmitirme tus ánimos y hacerme reír tanto. Porque eres especial y espero que el karma te devuelva pronto lo que te debe.

Y a César por ser mi primer guía, gracias al que adquirí grandes conocimientos y porque intentaste hacer un ingeniero de mí.

Resumen

Debido a los problemas en la gestión de la información sobre interacción entre fármacos y con ello los problemas en la seguridad de los pacientes, en este proyecto se desarrolla un sistema para la extracción de interacciones entre medicamentos sobre textos biomédicos para que los profesionales de la biomedicina tengan un rápido acceso a dichas información. Partiendo del trabajo realizado en la tesis "*Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions*." *Segura, I.* (1), nos basamos en su corpus DrugDDI para evaluar nuestro enfoque mediante técnicas de aprendizaje automático basadas en características con el algoritmo SMO (SVM).

Palabras clave: interacción entre fármacos, IF, detección de relaciones, aprendizaje automático, SVM, SMO.

Abstract

Due to problems in the management of information about drug interaction and thus the problems with patient safety, this project develops a system for extraction of drug interactions on biomedical texts for practitioners of biomedicine, so they have quick access to such information. Based on work in the thesis "*Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions*." *Segura, I.* (1), we rely on the corpus DrugDDI to asses our approach using machine learning techniques based on features with the SMO algorithm (SVM).

Keywords: drug-drug interactions, DDI, relation extraction, machine learning, SVM, SMO.

Contenido

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

Capítulo 1. Introducción

Las reacciones adversas a medicamentos (RAM) son un hecho relevante en el tratamiento de enfermedades debido a su incidencia y, que como consecuencia, acarrean riesgo de mortalidad y un alto coste. Según un estudio epidemiológico realizado en Reino Unido (2), en torno al 6,5% de los pacientes atendidos sufrieron algún tipo de RAM. Las reacciones adversas pasan frecuentemente inadvertidas para el médico y pueden evitarse en más de la mitad de los casos (3). Dentro de la mayoría de casos evitables, y ocupando un 16,6% de las RAM del anterior estudio, nos encontramos con pacientes que sufrieron una interacción farmacológica (IF). Las IF corresponden a la modificación de los efectos de un fármaco en presencia de otro, pudiendo inhibir un fármaco al otro, aumentar la toxicidad o disminuir su eficacia. En otro estudio sobre 756.047 pacientes durante 4 años (4), se publica que el porcentaje de pacientes con tratamientos crónicos que sufrieron al menos un IF se elevó entre un 6,2% y un 6,7%. Además se situó entre el 2,0% y el 2,3% para los usuarios del plan de salud.

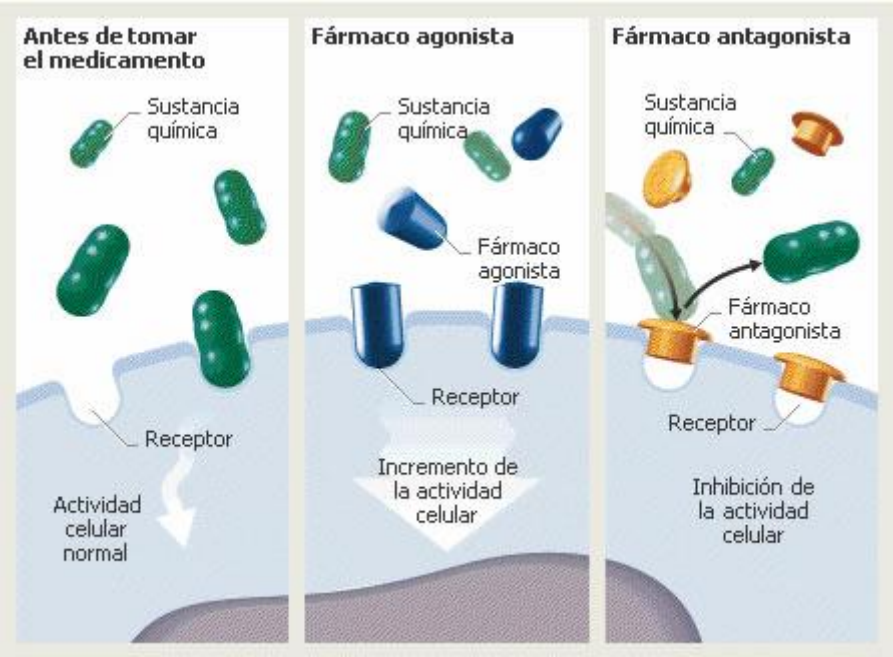


Ilustración 1 Interacción entre fármacos (Fuente: http://www.miportal.edu.sv)

Aunque se disponen de varias bases de datos que contienen interacciones entre medicamentos, éstas se hayan incompletas. La OMS colabora en la recolección de estos datos mediante el Programa OMS de Vigilancia Farmacéutica Internacional en el que colaboran profesionales sanitarios a nivel mundial. Esto desemboca en que el personal sanitario debe revisar numerosos documentos biomédicos que se refieren a la seguridad de medicamentos para poder tener información más actualizada. La cantidad ingente de información al respecto repercute en la saturación de los profesionales ante esta avalancha de datos que crecen constantemente (1). Sería de gran ayuda que este

proceso de recolección de información se produjera pues de forma automática, de forma que esta información vital fuera accesible de forma inmediata.

El hecho de poder contar con un método automático para la extracción de este tipo de información mejoraría la detección temprana de interacciones farmacológicas y ayudaría reducir las muertes provocadas por IF. Además, ayudaría a reducir el tiempo empleado por los profesionales sanitarios en la revisión de la literatura biomédica para esta detección y el coste asociado a la atención de los pacientes que sufren una IF (en el estudio realizado en (2) el coste anual para el *National Health Service* ascendía a 706M€). La tesis doctoral "*Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions*." *Segura, I.* (1) propone el desarrollo de técnicas de extracción de información aplicadas al dominio farmacológico, en particular, para la detección automática de IF en textos biomédicos.

Continuando con su estudio, en este proyecto tratamos la continuación a esta aportación, esta vez con el estudio de otras técnicas de aprendizaje automático, en concreto *Support Vector Machines* (SVM) basado en características.

Objetivos

Es por ello que el objetivo principal se enfoca en desarrollar un sistema basado en SVM que permita detectar interacciones farmacológicas, utilizando el Corpus *DrugDDI* (1) anotado con 3.160 IF. Las características de las oraciones del corpus son utilizadas junto con información obtenida de recursos externos para entrenar el clasificador SVM, en concreto su implementación de *Sequential Minimal Optimization* (SMO) por sus buenos resultados en la extracción de relaciones en el dominio biomédico. Finalmente realizaremos la comparación con los métodos propuestos en la tesis de Segura I. para analizar si existen mejoras sobre sus resultados.

Para poder llevar a cabo nuestro objetivo principal, nos proponemos una serie de objetivos que se centran en alcanzar el conocimiento previo necesario para desarrollar el sistema y la posterior evaluación. En concreto, proponemos los siguientes objetivos específicos:

- Comprender la necesidad del tratamiento automático de los textos biomédicos en el dominio farmacológico.
- Conocer los sistemas de extracción de relaciones y su evaluación.
- Estudiar las principales líneas de investigación en la extracción de relaciones en el dominio biomédico.
- Estudiar la herramienta *Weka* para la aplicación de técnicas de aprendizaje automático para el tratamiento de los textos.

- Estudiar herramientas y recursos para el procesamiento del lenguaje natural en el dominio biomédico, en particular, el estudio de *UMLS* y *MMTx*.
- Estudiar la clasificación ATC para los fármacos y desarrollar una herramienta para la clasificación automática que se apoye en dicha clasificación.
- Estudiar herramientas que ayuden a extraer información de la Web: *crawlers* y *robots*. En particular las herramientas *OpenKapow RoboMaker* y *Web-Harvest*.
- Estudiar de forma detallada el trabajo desarrollado en la tesis "*Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions*."*Segura, I.* (1)
- Estudiar las principales características del corpus *DrugDDI* (5).
- Analizar las alternativas para el tratamiento de los ficheros XML del corpus.
- Estudiar metodologías para el desarrollo que agilicen nuestro trabajo: en concreto *Scrum* y *eXtreme Programming*.
- Implementar un método basado en Castor para la extracción de características de los ficheros XML, las cuales útiles en la extracción de interacciones farmacológicas.
- Realizar experimentos con el algoritmo SMO y distintos conjuntos de características.
- Comparar los resultados obtenidos por el algoritmo SMO con los obtenidos en la tesis (1) que sirve como base para este trabajo.

Estructura del documento

Para llevar a cabo este propósito, presentamos el estudio sobre el estado del arte en el *Capítulo 2*. En este estudio nos centramos en *La extracción de relaciones* ya que la extracción de relaciones forma parte de la base del sistema. A continuación presentamos la *Evaluación de sistemas de extracción de información en biomedicina* ya que nos servirán para medir los resultados de nuestro sistema. Y después de estos puntos, procedemos a analizar los *Principales sistemas de extracción de relaciones en biomedicina*; este apartado nos da la visión general para conocer la importancia de los sistemas de extracción de relaciones en el dominio biomédico y como consecuencia de este estudio elegimos las técnicas de aprendizaje automático para aplicarlas en la extracción de relaciones biomédicas. Es por ello que el siguiente apartado se centra en el estudio de los sistemas de *Aprendizaje automático*: centrándonos en la herramienta Weka (6) (7) (8) y en el algoritmo SMO. Para finalizar este capítulo, estudiaremos la *Metodología de desarrollo ágil* que nos servirá para desarrollar el sistema de extracción de relaciones de forma flexible y adaptándonos a las futuras necesidades.

El paso siguiente dentro de este documento, correspondiente al Capítulo 3, es la aplicación de la clasificación mediante *SVM aplicado a la extracción de interacciones*

farmacológicas en textos biomédicos. Para poder realizar la extracción, primeramente necesitamos conocer *El corpus DrugDDI* ya que será la base del sistema y, gracias a sus anotaciones, podremos entrenar el algoritmo SMO y evaluar fácilmente sus resultados. A continuación explicaremos cómo hemos procedido al *Desarrollo del sistema*, recogiendo atributos tanto del corpus como de recursos farmacológicos externos como DrugBank o ATC System. También conoceremos los datos sobre la integración de utilidades para el manejo de los ficheros XML. A continuación presentaremos los resultados de los *Experimentos* realizados y, finalmente, presentaremos en la sección *Discusión* una comparativa entre los resultados obtenidos por el método Kernels presentado en (1) y el método aplicado en este proyecto para conocer si existen mejoras en la aplicación de SMO.

Para concluir el documento, en el Capítulo 4 procederemos a exponer nuestras conclusiones generales y nuestra visión para trabajos futuros al respecto.

Capítulo 2. Estado del arte

Este capítulo se centra en adquirir los conocimientos básicos para poder llevar a cabo la implementación y evaluación de nuestro sistema de extracción de interacciones entre medicamentos.

Para conseguir el objetivo de este capítulo empezamos estudiando en qué consiste la extracción de relaciones y nos centramos en los principales sistemas de extracción de relaciones en el dominio biomédico (que es en el que nos encontramos): técnicas de lingüística, de patrones y de aprendizaje automático. A continuación estudiamos con más detenimiento las técnicas de aprendizaje automático debido a los buenos resultados en el dominio biomédico. En concreto, estudiamos los métodos basados en características (los ejemplos son representados como vectores de características o atributos) para poder contrastar los resultados con los obtenidos en la tesis (1) basada en métodos *kernels*. Con el fin de poder realizar la evaluación de la propuesta de este proyecto y su comparación con los resultados obtenidos en (1), también estudiamos las principales métricas utilizadas en la evaluación de los sistemas de extracción de información.

Abordamos a continuación qué es el aprendizaje automático, lo cual nos ayuda a asentar los conocimientos necesarios para poder realizar nuestros experimentos. Para ello estudiamos con más detalle la herramienta de aprendizaje automático Weka y el algoritmo SMO que usamos para la extracción de relaciones entre fármacos.

Finalmente, procedemos a estudiar metodologías ágiles, descubriendo qué nos pueden aportar *Scrum* y *eXtreme Programming* en el desarrollo de nuestro proyecto. Enlazando con los hitos correspondientes a nuestra metodología, describimos las principales tecnologías que se han incorporado a lo largo del desarrollo de nuestro prototipo.

La extracción de relaciones

Con la extracción de relaciones lo que se persigue es poder realizar la detección de relaciones semánticas entre entidades en un texto. Esta tarea está presente en multitud de sistemas de recuperación de información y sistemas de pregunta-respuesta sobre diferentes dominios. La mayoría de los sistemas de extracción de relaciones se centran en la extracción de relaciones binarias; es decir, relaciones uno a uno. Sin embargo, algunos sistemas de extracción de relaciones pueden llegar a detectar múltiples relaciones semánticas: este hecho se da, por ejemplo, en la extracción de relaciones en el análisis de vídeos, en el que la compleja estructura de un vídeo puede necesitar la extracción de relaciones semánticas más complejas para su posterior clasificación con la finalidad de salvar el vacío semántico (9).

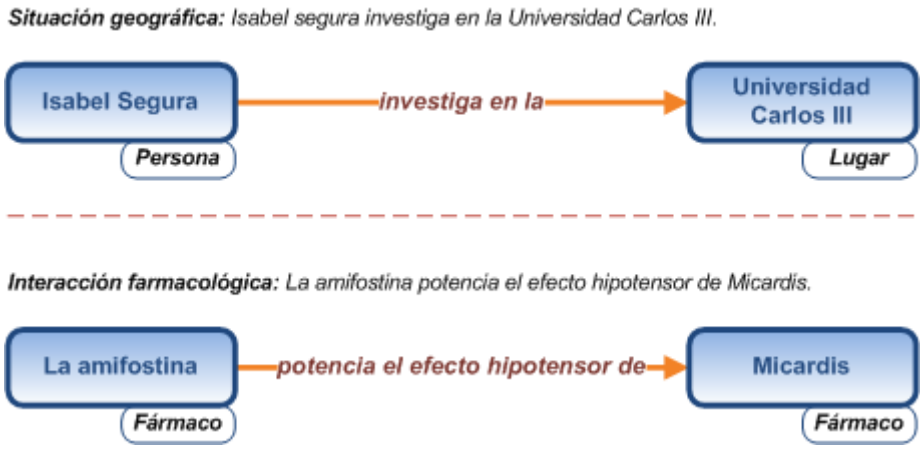


Ilustración 2 Ejemplos de relaciones

Dentro del dominio biomédico la extracción de relaciones binarias se puede aplicar para poder detectar relaciones como la interacción proteína-proteína (IPP) o interacción fármaco-fármaco (IFF) .

Para poder extraer relaciones, se necesitan varios procesos de análisis del lenguaje como pueden ser la extracción de raíces de las palabras, la derivación, el análisis sintáctico y semántico de las frases, la desambiguación y la anáfora. A modo de resumen podemos destacar algunos de los recursos más útiles en la extracción de relaciones (10):

- **Información del contexto:** los elementos que rodean o están entre las dos entidades pueden aportar mayor información de cara a verificar que existe una relación entre ambas entidades.
- **Part-of-speech tagging (POST):** gracias a la anotación de los elementos de la oración en su respectiva categoría podemos identificar más fácilmente las entidades (que suelen corresponderse con aquellos elementos etiquetados como nombres o sintagmas nominales) y las relaciones entre las entidades (que suelen encontrarse etiquetadas como verbos).
- **Árboles de análisis sintáctico completo:** este tipo de análisis aporta mayor información que el POST, de manera que ayuda a entender las relaciones entre las entidades mediante la agrupación de las palabras en una jerarquía (como el sintagma nominal, el sintagma verbal y el sintagma preposicional). No obstante, este proceso es muy costoso y consume mucho tiempo.
- **Grafos de dependencia:** son la alternativa al análisis sintáctico. Este tipo de grafo representa las relaciones entre las palabras de una oración, de manera que genera un grafo que enlaza cada palabra con las palabras que dependen de ella.

Volviendo al dominio biomédico, la pretensión es encontrar alguna relación predefinida entre un par de entidades como pueden ser la interacción entre proteínas, la interacción entre fármacos, la interacción entre hierbas y medicamentos y otras relaciones de interés en el dominio biomédico. La mayoría de los grupos de investigación

se han centrado en la extracción de proteínas, aprovechado la existencia de corpus anotados con este tipo de información biológica. El hecho de que no existan corpus anotados o la calidad de los mismos sea menor en otros ámbitos, unido al hecho de que se usen distintos corpus en las investigaciones y se traten distintos tipos de relaciones, hace que el análisis y comparación de los resultados obtenidos sea una tarea compleja.

Principales sistemas de extracción de relaciones en biomedicina

Actualmente, los mayores aportes a la extracción de relaciones en biomedicina se centran en tres categorías (11):

- **Basados en lingüística:** para este tipo de extracción se utilizan tecnologías de ingeniería lingüística. Para ello se realiza el procesado de textos en lenguaje natural analizándolos sintácticamente y semánticamente intentando descubrir relaciones. Este método se divide en dos tipos en función de la complejidad de las técnicas usadas: análisis superficial y análisis profundo.
- **Basados en patrones:** mediante una serie de reglas específicas del dominio se extraen aquellas relaciones que encajan con éstas.
- **Basados en aprendizaje automático:** mientras que en las categorías anteriores se precisa de diferentes conjuntos de gramáticas y reglas, la extracción mediante aprendizaje permite que la adquisición de conocimiento sea automática.

Métodos basados en aprendizaje automático:

El enfoque que encontramos para tratar la extracción de relaciones se transforma en una tarea de clasificación empleando algoritmos de aprendizaje automático. Esto ha fomentado el uso de estos métodos con preferencia frente a los métodos basados en patrones dado que en estos últimos es necesaria la codificación manual del conocimiento para poder generar las reglas. Sin embargo, los métodos de aprendizaje automático no requieren de esta codificación previa pues son capaces de extraer el conocimiento de forma automática. Otra ventaja en los métodos de aprendizaje automático es que son fáciles de migrar a otros dominios o conjunto de datos, mientras que en los basados en patrones habría que redefinirlos. Sin embargo no todo son ventajas ya que los algoritmos de aprendizaje automático ocasionan un gran gasto computacional en el entrenamiento y en las pruebas sobre grandes volúmenes de información. Se deben construir representaciones que definan tanto casos positivos como negativos. Dependiendo de la forma de representar estos datos nos encontramos con dos categorías: basados en atributos y basados en *kernels*.

- ***Métodos basados en atributos (features):*** se extraen las características del fichero de entrada y se representa cada instancia (ejemplo) como un vector de

características. Con un conjunto de instancias se entrena la máquina de aprendizaje automático para poder clasificar otras instancias que no hayan formado parte del entrenamiento.

- ***Métodos basados en funciones kernels:*** la representación de las instancias no se realiza en forma de vector sino que pueden ser representadas por medio de estructuras tales como grafos de dependencia, secuencia de palabras o árbol de análisis sintáctico. Se requiere una función *kernel* para poder medir la similitud entre dos estructuras. Por medio de esta función *kernel*, es posible determinar la similitud entre una nueva instancia y un ejemplo del corpus de entrenamiento.

Si atendemos a distintos sistemas que usen estos métodos y analizamos las últimas investigaciones de los últimos años, podemos ver que existe una gran cantidad de sistemas que emplean aprendizaje automático.

Por ejemplo, en el 2005 (12) nos encontramos con la clasificación de documentos de MEDLINE intentando encontrar aquellos que contienen interacciones farmacológicas para saber si un documento contiene o no interacciones mediante la aplicación de SVM. Para ello recurre a la ayuda de *MetaMap* (MMTx) como herramienta de ingeniería lingüística para aplicar relaciones sobre *Unified Medical Language System* (UMLS).

En el 2007 nos encontramos con la investigación (13) de un método para extraer oraciones que contengan también interacciones entre proteínas. El problema se enfoca mediante la creación de árboles de dependencia en el que se mide la similitud a través de la distancia en los caminos entre los nombres de las proteínas. Para ello se contempla el uso de varios algoritmos de aprendizaje: SVM, *transductive* SVM, funciones harmónicas y *k-nearest-neighbor*, obteniendo mediante SVM los mejores resultados con una medida-F: 85,22%.

Podemos encontrar otro enfoque distinto en el 2008 (14), en donde se construye un sistema para la detección de interacciones entre proteínas mediante análisis sintáctico combinado con grafos *kernel* mediante el algoritmo *Sparse RLS*.

Otra aproximación (15) propone distintas técnicas de procesado del lenguaje natural sobre resúmenes de documentos biomédicos para encontrar interacciones entre proteínas. AIMed es el dataset usado ya que contiene estos abstracts convenientemente anotados. Se aplica análisis de dependencias, análisis profundo y análisis de la estructura de las oraciones. Finalmente se aplica SVM sobre los atributos extraídos con una medida-F del 59,5%.

En BioPPISVMExtractor del 2009 (16) también intenta extraer interacciones entre proteínas. Para ello recurre a clasificación mediante SVM, analizando tanto palabras, palabras clave, nombres y distancias de proteínas, y el camino Link (se añaden atributos

relacionados con Link Grammar para mejorar la precisión). Todo ello se prueba sobre el corpus DIP con una medida-F del 47,53%.

Dado que los métodos de aprendizaje automático consiguen aprender relaciones nuevas a partir de un conjunto de entrenamiento con buenos resultados en la extracción de relaciones semánticas (17), encaminaremos nuestro proyecto para trabajar con ellos. En la tesis sobre la que basamos nuestro trabajo se emplean funciones *kernel*, de manera que nos centraremos en el estudio de los algoritmos basados en atributos, para su posterior comparación con los métodos *kernels* propuestos en (1). Antes de llevar a cabo este estudio, revisaremos los métodos que se usan para evaluar los sistemas de extracción en biomedicina, ya que con ellos podremos evaluar nuestro sistema.

Evaluación de sistemas de extracción de información en biomedicina

Uno de los problemas fundamentales en cualquier sistema de extracción de información es su evaluación. Para poder evaluar los resultados obtenidos debemos seleccionar unas métricas que sean claras, reproducibles y fáciles de entender. Dado que en nuestro sistema partimos de colección de datos que dividimos en un conjunto de entrenamiento y en un conjunto de test. A partir del conjunto de entrenamiento el sistema deberá inferir las categorías de los ejemplos del conjunto de test. Puesto que las categorías de estos ejemplos son conocidas, podremos evaluar las inferencias del sistema tomando como base la matriz de confusión en la que nos encontramos con las cuatro posibilidades a las que puede optar la anotación del sistema (por encontrarnos con un problema de clasificación binaria):

- **Verdadero positivo:** son aquellos elementos que son positivos y se anotan correctamente como positivos.
- **Falso positivo:** son aquellos elementos que el sistema detecta como positivos aunque en realidad sean negativos.
- **Verdadero negativo:** se anota como verdaderos negativos a aquellos elementos que el sistema reconoce correctamente como negativos.
- **Falso negativo:** bajo este nombre denominamos a aquellos elementos que, aún siendo positivos, se anotan incorrectamente como negativos.

		RESULTADO CORRECTO	
		Positivo	Negativo
RESULTADO INFERIDO	Positivo	VP (Verdadero Positivo)	FN (Falso negativo)
	Negativo	FP (Falso Positivo)	VN (Verdadero negativo)

Tabla 1 Matriz de confusión

Los falsos negativos y los verdaderos positivos están relacionados dado que ambos son positivos en la realidad. Lo mismo ocurre con los falsos positivos y los verdaderos negativos. De esta manera podemos denominar como N^+ al número de positivos reales en el sistema, es decir, a la suma de los verdaderos positivos con los falsos negativos:

$$N^+ = VP + FN$$

Ecuación 1 Número de positivos reales

De la misma forma podemos definir como N^- al número de negativos reales en el sistema, o lo que es lo mismo, la suma de verdaderos negativos con los falsos positivos:

$$N^- = VN + FN$$

Ecuación 2 Número de negativos reales

Y a partir de estas ecuaciones básicas nos encontramos con una serie de funciones que nos sirven como medidas de evaluación de un sistema de categorización.

Sensibilidad o Cobertura o Recuerdo o Tasa de Verdaderos Positivos (TVP o Recall): es la porción de verdaderos positivos predichos entre todos los positivos posibles. Es decir, es la probabilidad de que si una instancia pertenece a una categoría, ésta instancia se clasifique con la categoría correcta. Es una medida muy usada en test médicos.

$$TVP = Cobertura = C = \frac{VP}{VP + FN} = \frac{VP}{N^+}$$

Ecuación 3 Tasa de verdaderos positives o cobertura

Tasa de Falsos Positivos (TFP): mide la fracción de ejemplos negativos que son clasificados incorrectamente como positivos.

$$TFP = \frac{FP}{VN + FP} = \frac{FP}{N^-}$$

Ecuación 4 Tasa de falsos positivos

Especificidad o tasa de verdaderos negativos (TVN): nos ofrece la proporción de instancias negativas correctamente clasificadas.

$$TVN = \frac{VN}{VN + FP} = \frac{VN}{N^-}$$

Ecuación 5 Tasa de verdaderos negativos o especifidad

Tasa de Falsos Negativos: proporciona la fracción de las instancias positivas que son clasificadas incorrectamente como negativas.

$$TFN = \frac{FN}{VP + FN} = \frac{FN}{N^+}$$

Ecuación 6 Tasa de falsos negativos

Precisión: es la proporción de instancias clasificadas como positivas correctamente frente al total de predichos para esa categoría, o lo que es lo mismo, es la probabilidad de que una instancia sea categorizada dentro de una categoría y realmente pertenezca a ella.

$$Precisión = P = \frac{VP}{VP + FP}$$

Ecuación 7 Precisión

Exactitud: fracción de las clasificaciones realizadas en las que se asigna correctamente la categoría.

$$Exactitud = \frac{VP + VN}{VP + FN + VN + FP} = \frac{VP}{N^+ + N^-}$$

Ecuación 8 Exactitud

Para la evaluación de un sistema de clasificación, las medidas que se suelen usar son la cobertura y la precisión. La precisión mide lo bueno que puede ser un sistema clasificando mientras que la cobertura muestra la habilidad del sistema para detectar las clasificaciones correctas existentes.

Un sistema perfecto sería aquel que tuviera FP = 0 y FN = 0. Nos encontramos con que la precisión y la cobertura están en contraposición y lo normal es que cuando una ascienda la otra descienda. Por ello, para poder evaluar el sistema harmónicamente necesitamos una medida que combine tanto la precisión como la cobertura, y para ello contamos con la medida-F.

$$F(\beta) = \frac{(1 + \beta^2)PC}{\beta^2 P + R}$$

Ecuación 9 Medida-F

Dentro de esta fórmula nos encontramos con β que es un valor positivo que aporta mayor peso o a la cobertura o a la precisión. Si $\beta > 1$ se da mayor importancia a la cobertura, y si $\beta < 1$ se da mayor peso a la precisión. En el caso de $\beta = 1$ ambos adquieren la misma importancia.

Así pues, tomaremos tres medidas típicas en biomedicina (11) para evaluar nuestro sistema de clasificación basado en aprendizaje automático: precisión, cobertura y medida-F (18).

Aprendizaje automático

El aprendizaje automático tiene por objetivo el desarrollo de técnicas, principalmente mediante métodos matemáticos, que permiten a un sistema informático inducir el conocimiento: a partir de la observación repetida de objetos con las mismas características se establece una conclusión para todos los objetos con la misma naturaleza.

Existen varios tipos de aprendizaje automático, entre ellos podemos destacar: el aprendizaje supervisado en el que la base de conocimiento se basa en función de la etiquetación de problemas anteriores (usado para clasificación de vectores, es el que usaremos en nuestro sistema ya que disponemos de estas anotaciones). Otro método es el aprendizaje no supervisado: sólo disponemos de un conjunto de ejemplos sin anotar su categoría y el sistema intenta agruparlos por similitud (19).

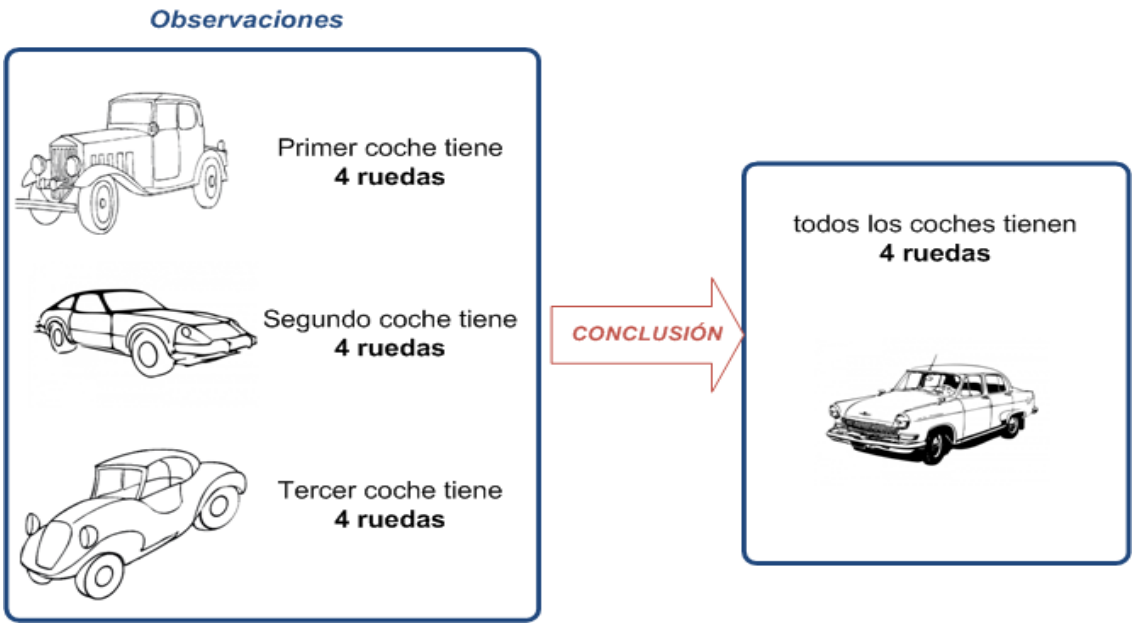


Ilustración 3 Inducción de conocimiento

El aprendizaje supervisado se usa fundamentalmente para extraer relaciones entre múltiples atributos y ha sido aplicado con éxito en esta tarea (17). Para poder usar estos algoritmos debemos seguir una serie de pasos:

- Extraer el conjunto de datos: lo ideal es que un experto del dominio nos indique cuáles pueden ser relevantes.
- Preparación de los datos y preprocesado: si la información extraída contiene ruido, podríamos disminuirlo preprocesando los datos. En este paso también se puede experimentar eliminando atributos irrelevantes o redundantes.
- Selección del algoritmo y evaluación: debemos evaluar qué algoritmo se ajusta mejor a nuestro conjunto de datos para poder realizar la clasificación. Para ello existen distintas técnicas como coger dos tercios de los datos para entrenamiento y un tercio para evaluación o cross-validation en el que se forman varios conjuntos de excluyentes de entrenamiento y de pruebas analizando al completo el conjunto de datos, de manera que se obtiene el error estimado del clasificador.

Dentro del aprendizaje supervisado nos encontramos con distintos métodos para encontrar un resultado satisfactorio, entre otros nos encontramos con los que están basados en árboles de decisión, los que se basan en reglas, los que lo hacen sobre redes neuronales, sobre algoritmos estadísticos y los que lo hacen sobre Support Vector Machines.

Basándonos en el estudio realizado en (17), nos decidimos por usar SVM ya que ofrece buenos resultados en múltiples dominios, incluyendo el de la biomedicina atendiendo a que tenemos suficientes datos para entrenar al SVM, tenemos instancias con muchas

dimensiones, ejemplos desbalanceados y nos hallamos ante un problema binario de clasificación.

En el siguiente apartado explicamos Weka, una de las herramientas más usadas para aplicar aprendizaje automático, debido a que es una de las herramientas más completas y antiguas, la extensa colección de algoritmos implementados y su facilidad de uso.

Weka (6) (7) (8)

El nombre proviene de Waikao Enviroment for Knowledge Analysis. Es un entorno software que facilita el análisis del conocimiento mediante aprendizaje automático y minería de datos, está desarrollado en Java (multiplataforma) por la Universidad de Waikato y es software libre bajo licencia GNU-GPL, con lo que el software es de libre distribución y difusión. La versión original se inició en 1993, su implementación era un front-end en TCL/TK que modelaba algoritmos implementados en otros lenguajes de programación y contenía una sería de utilidades de preprocesamiento implementadas en C para realizar experimentos de aprendizaje automático. En un principio la herramienta fue pensada para analizar datos procedentes del dominio de la agricultura, sin embargo, actualmente es usada en diferentes ámbitos con motivo de diferentes investigaciones.

En su actual versión ofrece un poderoso entorno con una colección de algoritmos para el análisis de los datos y el modelado predictivo. Además ofrece una interfaz gráfica de usuario que permite acceder cómodamente a sus funcionalidades. Dentro de las tareas que facilita este paquete tenemos el preprocesamiento de datos, clustering, clasificación, regresión, visualización y selección. En el caso de necesitar algo más, Weka está diseñado como una herramienta fácil de extender, por lo que es sencillo añadir nuevas funcionalidades o algoritmos.

El acceso a los datos para su estudio se produce mediante un fichero plano *Attribute-Relation File Format (*ARFF) en el que cada registro de datos se describe por un número prefijado de atributos. No obstante, se proporciona acceso a diferentes fuentes como a bases de datos vía SQL mediante conexión Java *Database Connectivity (*JDBC) procesando las consultas realizadas a la base de datos, ficheros *Comma Separated Values (*CSV), ficheros con instancias serializadas y otras fuentes de información. No puede realizar minería de datos multi-relacional, aunque existen herramientas que pueden convertir una colección de tablas relacionadas con una base de datos en una única tabla para ser procesada por Weka (20).

La salida de nuestro sistema será en formato CSV por la facilidad en cuanto a la generación. CSV es un formato abierto y sencillo usado para representar datos en forma de tabla. Las columnas de la tabla se corresponden a los datos separados por comas (o punto y coma en el caso de que nos encontremos en un país en el que el separador

decimal sea la coma). Las filas se separan mediante saltos de línea. No obstante, para realizar los distintos preprocesados de los datos usaremos el formato ARFF, de manera que podremos eliminar atributos o modificar su tipo si así lo creyéramos oportuno antes de pasar a su evaluación aplicando distintos filtros.

El formato ARFF

El formato nativo de Weka es el formato ARFF, como hemos dicho antes, en él se definen los registros mediante instancias con sus atributos. Consideramos instancia a cada grupo de características pertenecientes a un ejemplo a evaluar. Cada instancia se clasificará en función de sus atributos, que son las características que pueden ayudar a categorizar a la instancia.

Este formato está compuesto por una estructura en la que podemos diferenciar tres partes:

- **Cabecera:** es donde se define el nombre de cada relación. El nombre deberá ir entrecomillado si existen espacios entre sus caracteres. La sintaxis sería la siguiente:

@relation <nombre-de-relación>

- **Declaraciones de atributos:** en esta parte de la estructura nos encontramos con la declaración de los atributos que conforman el archivo y el tipo correspondiente a estos atributos. Weka considera varios tipos como pueden ser NUMERIC (números reales), INTEGER (números enteros), DATE (fechas en las que se define el patrón mediante una etiqueta entrecomillada siguiendo la nomenclatura de Java para las definiciones: dd→ día, MM→ mes,…), STRING (cadenas de texto que deben ir entrecomilladas si contienen espacios), ENUMERADO (se predefinen los valores aceptados del mismo modo que se

@attribute <nombre-de-atributo> <tipo>

declaran arrays de cadenas en java, es decir, el tipo serían el conjunto de valores aceptados entre corchetes). Se define la sintaxis de la siguiente forma:

- **Sección de datos:** se declaran los datos que componen la relación separando entre comas los atributos y entre saltos de línea las relaciones. Existe también un formato abreviado útil para cuando hay muchos atributos en una relación que contienen el 0, de forma que éstos se ignoran y sólo se escriben los relevantes precedidos por su posición. A continuación presentamos la sintaxis para una

relación con varios atributos, en su formato completo a la izquierda y en su formato abreviado a la derecha.

```
@data
4,0,0,0,0,3
```

```
@data
0 4,5 3
```

En el caso de querer transformar los datos, por ejemplo eliminar atributos que no sean relevantes para la clasificación, podemos preprocesar los datos recurriendo al uso de filtros que modifiquen nuestros datos y eliminen aquellos datos que no se consideren relevantes o que sólo aporten ruido.

Filtros

Dentro del preprocesado nos encontramos con la aplicación de diversos filtros que nos permiten realizar transformaciones de todo tipo sobre los datos. Entre ellos podemos mencionar algunos relacionados con el preprocesado de los atributos como el *AddExpression* que permite agregar un atributo que se corresponde con el valor de una función que se puede calcular a partir de otros atributos, esto nos sería útil si quisiéramos usar una ecuación que fusionara varios atributos. Por su parte, el filtro *Discretize* discretiza un conjunto de valores numéricos en rangos de datos; por ejemplo, podríamos simplificar un atributo que recogiera valores analógicos convirtiéndolos en digitales, de manera que convertimos un atributo que podría tomar infinitos valores en un valor que tome un número finito de valores. Mediante el filtro *NominalToBinary* se produce una transformación de los valores nominales de un atributo en un vector cuyas coordenadas son binarias; este filtro puede resultar útil ya que algunos algoritmos no admiten datos nominales. *Remove* elimina el conjunto de atributos seleccionados, con lo que podemos eliminar los menos relevantes.

Por otra parte, tenemos también filtros relacionados con las instancias. Por ejemplo, disponemos del filtro *NonSparseToSparse* que convierte una muestra de modo completado al modo abreviado reduciendo el tamaño del fichero. *RemoveMisclassfied* elimina el conjunto de instancias mal clasificadas, lo cual dejaría en el sistema sólo instancias bien clasificadas como ejemplo. Y entre otros filtros, contamos también con *Resample* cuya misión es coger un conjunto aleatorio sobre el conjunto total, lo cual puede venir muy bien si nuestro conjunto de datos inicial es muy amplio para reducir los tiempos de entrenamiento.

Algortimos de clasificación

Como hemos dicho anteriormente, Weka dispone de diferentes herramientas para el análisis de datos y aprendizaje automático. Dentro de ellas nos encontramos con el clasificador que agrupa tanto algoritmos de clasificación como de regresión dentro del aprendizaje supervisado para estimar la precisión del modelo predictivo. El más antiguo de estos modelos de aprendizaje es el *ZeroR.* El siguiente fue el *OneR* (21) y también forma parte de los algoritmos basados en árboles de decisión, está basado en un árbol de un nivel de profundidad como única regla de decisión y selecciona la clase más frecuente para el atributo con menor error cuadrático. *NaiveBayes*, por su parte, implementa el clasificador probabilístico *Naive Bayesian*. *DecisionTable* emplea el método *wrapper* para encontrar un subconjunto de atributos a incluir en la tabla usando la búsqueda por el mejor primero. *IBk* se basa en la implementación de los k vecinos más cercanos (*k-nearest-reighbours*) (22), es un algoritmo de aprendizaje tardío o perezoso ya que no se realiza ninguna abstracción en forma de reglas o árboles de decisión, de esta forma la instancia se clasifica en la clase más frecuente a la que pertenecen sus K vecinos más cercanos. También nos encontramos con j48 que es una implementación de C4.5 (23) y produce árboles de decisión, este algoritmo incorpora la poda del árbol de clasificación una vez que éste se ha construido, de manera que se eliminan las ramas que aportan menor capacidad predictiva. Este último es un algoritmo estándar y es muy usado dentro del aprendizaje automático. SM0 implementa el algoritmo de optimización mínima secuencial para máquina de soporte vectorial (SVM) (24) y lo analizaremos con más detenimiento en la siguiente sección.

Nos encontramos con otros modelos de aprendizaje como son los meta-clasificadores que mejoran o extienden los modelos básicos de aprendizaje automático basándose en la salida de los otros modelos (*AdaBoostM1, MultiClassClassifer,…*). Además disponemos de modelos basados en reglas de asociación (*Apriori*) y *clustering* que sirven para dividir los datos en grupos o *clusters* y son usados dentro del aprendizaje no supervisado (algoritmo EM).

Bien, como hemos mencionado en el apartado con los principales sistemas de extracción de relaciones en biomedicina, los algoritmos SVM ofrecen buenos resultados en este problema de clasificación. Si vemos la siguiente tabla comparativa que aparece en (17) podemos ver cómo los mejores resultados se obtienen mediante SVM (en las investigaciones en biomedicina nos encontramos con buenos resultados igualmente en la extracción de relaciones (16) (25) (26)).

	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule-learners
Accuracy in general	**	***	*	**	****	**
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***	**
Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)	***(not directly continuous)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

Tabla 2Comparativa de algoritmos supervisados (=Malo;****=Bueno)*

Por estos buenos resultados es por la que escogemos SVM para nuestra clasificación. Como podemos observar, una de las desventajas es la velocidad de aprendizaje, para intentar mitigar esto usaremos la implementación SMO por ser una de más rápidas dentro de SVM (27).

SMO (7)

Este algoritmo pertenece a la categoría de algoritmos basados en ecuaciones matemáticas. Es capaz de aprender a partir de ejemplos generalizando sin tener que formalizar el conocimiento adquirido. Éste es un algoritmo muy eficiente especialmente en la categorización de texto (28). SMO implementa el algoritmo de entrenamiento para clasificadores SVM mediante núcleos polinomiales o gaussianos (29) (30).

El modelo SVM representa los puntos de la muestra en el espacio n-dimensional (conjunto de hiperplanos), con ello, en función de la cercanía se puede realizar la clasificación en la clase correspondiente. Si existe una buena separación entre las clases se permitirá una clasificación correcta a la hora de inferir la clase correspondiente a una instancia.

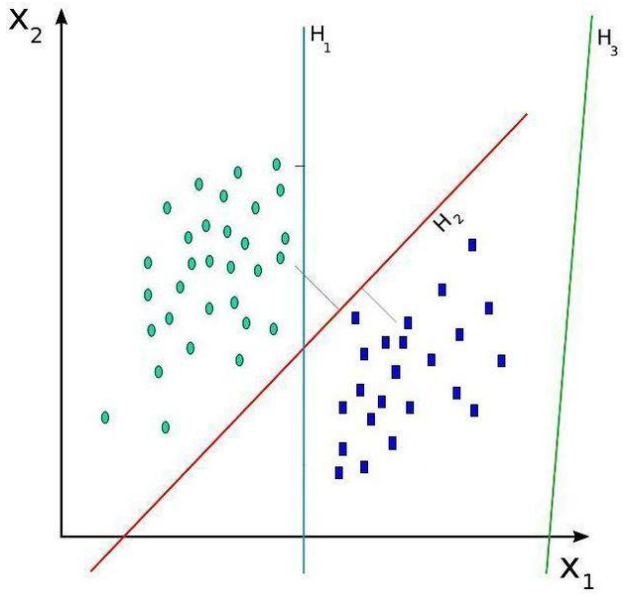


Ilustración 4 Ejemplo de hiperplano (una línea en este caso) que une las variables predictoras y define la frontera entre categorías sobre 2 dimensiones

Los clasificadores SVM poseen un entrenamiento muy eficiente, la clasificación también es muy eficiente, tiene un buen funcionamiento con problemas típicos y son extremadamente robustos para generalización con menos necesidad de heurísticos para entrenamiento. Por ello es resistente al sobreajuste; el hiperplano de margen máximo es relativamente estable, sólo depende de los vectores de soporte (añadir o eliminar instancias sólo afecta si son vectores de soporte).

Hay que destacar que este algoritmo divide el problema de Optimización Cuadrática en subproblemas Pequeños (OCP) que se resuelven analíticamente, evitando las iteraciones internas de los algoritmos de optimización. Presenta memoria lineal con el conjunto de entrenamiento (frente cuadrático). El tiempo es entre lineal y cuadrático. Y es muy eficiente si existen datos dispersos. Los clasificadores SVM son muy precisos en muchos dominios de aplicación y resistentes al sobreajuste. Se usan en reconocimiento facial, *Optical Character Recognition* (OCR), bioinformática, minería de texto y series temporales.

Los valores que no existen son reemplazados globalmente, los atributos de tipo nominales se transforman en tipos binarios y los atributos se normalizan por defecto ya que los coeficientes en la salida dependen de datos normalizados. Se puede desactivar la normalización, o la entrada se puede normalizar mediante media 0 y varianza. Cuando se trabaja con instancias dispersas, se puede desactivar la normalización para aumentar la eficiencia del algoritmo y consumir menor tiempo. Se pueden ajustar los modelos de regresión logística en la salida del SVM de cara a obtener estimaciones probabilísticas. El SMOreg implementa el algoritmo de optimización mínima secuencial para problemas de regresión (31).

A continuación presentamos una tabla con algunas de las ventajas y desventajas (17) más importantes para este algoritmo:

Ventajas	Inconvenientes
• Muy buenos resultados en clasificación. • La velocidad de clasificación es muy buena. • Alta tolerancia a valores irrelevantes. • Robustos ante valores redundantes o dependientes.	• La velocidad de aprendizaje respecto al número de atributos y de instancias es lenta aunque mejor que en el resto de SVM. • No trata atributos discretos. • La explicación de las clasificaciones o transparencia del conocimiento es pobre.

Tabla 3 Ventajas y desventajas de SMO

Metodología de desarrollo ágil

Para la ejecución del proyecto hemos seguido, dentro de lo posible, metodologías ágiles tanto en la planificación como en el desarrollo. Las metodologías ágiles son útiles en el desarrollo de software ya que se basan en procesos iterativos y evolutivos, los cuales se llevan en un entorno de máxima colaboración, de manera que el resultado final es un software de alta calidad que satisface las necesidades cambiantes del cliente. Esto ocurre, a diferencia de las metodologías tradicionales, ya que se pone más énfasis en la adaptabilidad que en la previsibilidad.

En concreto, nos basamos en *Scrum* para las mejores prácticas en la gestión del proyecto y en *eXtreme Programming* (XP) para las mejores prácticas de programación.

eXtreme Programming (32)

Simplemente mencionar los principales valores en los que se fundamenta esta filosofía, los cuales se seguirán a lo largo del desarrollo del proyecto:

- **Simplicidad:** simplificación de diseño para facilitar el desarrollo y el mantenimiento.
- **Comunicación continua:** en todos los niveles, entre el equipo de desarrollo y con el cliente.
- **Retroalimentación:** debido a la comunicación con el cliente éste nutres con su opinión de manera que es fácil adaptarse en cada iteración sin tener que llevar a cabo grandes modificaciones.
- **Coraje y valentía:** algunas de las características fundamentales requieren de estos valores, como la programación por parejas, que puede ser considerada como una pérdida de tiempo por los gerentes o jefes. O no caer en el desarrollo

directo de infraestructuras para uso futuro directamente sino abordarlas mediante refactorización del código en las siguientes iteraciones.

- **Respeto:** respeto mutuo del trabajo entre miembros del equipo y orientación a mejorar el trabajo de todos.

Metodología Scrum (33)

Scrum surge a raíz de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Tekeuchi e Ikujijo Nonaka a mediados de los 80. Empezó siendo usado en el desarrollo de productos tecnológicos pero resulta válido para cualquier dominio en el que los requisitos sean inestables, necesiten rapidez y flexibilidad. En la década de los 90 fue cuando aparecieron los primeros artículos públicos (34) describiendo ésta metodología y también en ésta década se aplicó por primera vez al software. En el año 2001, Schwaber y Mike Beedle describieron la metodología orientada a desarrollo software en su libro *Agile Software Development with Scrum*.

Scrum es pues un método de trabajo, orientado principalmente a la gestión del proyecto y cuyas premisas se basan en que los requisitos del proyecto pueden cambiar a lo largo del mismo. Es por esto que se acepta que no se puede comprender completamente el problema definido desde un principio para hacer una planificación, sino que hay que centrarse en que se pueden producir modificaciones y se debe maximizar la capacidad del equipo para atender a los nuevos requisitos que puedan surgir respondiendo eficientemente. Es decir, se debe producir la adaptación continua a las necesidades de evolución del proyecto.

Los componentes principales son: roles, componentes del proceso, reuniones y Sprint. Los roles que se aplican al desarrollo software son 3: *Product Owner, Team y Scrum Master* (en el Scrum original son 6). Los componentes de proceso son: *Product Backlog, Sprint Backlog* e *Impediments Backlog*. También existen reuniones: *Sprint Planning, Daily Meeting, Sprint Review* y *Retrospective*. Y el *Sprint*, que es el proceso cíclico que sigue la metodología Scrum.

Roles

Product Owner: representa a los usuarios interesados en el producto. Suele pertenecer a la empresa que contrata el proyecto aunque también puede ser asumido por una persona del equipo de desarrollo. Es el responsable de aportar la visión de negocio.

Team o Equipo de Proyecto: es el equipo que desarrolla el proyecto, debe ser un equipo autónomo, autogestionado y multidisciplinar.

Scrum Master: es el encargado de supervisar la productividad del equipo y de conseguir hacer avanzar el proyecto en el caso de que se produzcan obstáculos. Suele ser asumido por el Jefe de Proyecto.

Detalles de la metodología

Partiendo del *Sprint 0* (el *Sprint* inicial), se obtiene la visión general del proyecto y el primer *Product Backlog* que contiene aquellas tareas priorizadas correspondientes a requisitos funcionales y no funcionales que el *Product Owner* y los *skateholders* esperan que contenga el producto. Además en este *Sprint* se realiza el *Estimation Meeting* que estima la totalidad del proyecto. A partir de este *Sprint* comienza el resto de *Sprints* de modo cíclico. En cada *Sprint* se realizan los siguientes pasos:

Sprint Planning: se definen los objetivos a los que se quiere llegar (*Selected Product Backlog*). De esta manera se repriorizan las tareas para cumplir con los objetivos obteniendo las tareas técnicas a realizar (*Sprint backlog*).

Daily meeting: Sincronización del equipo con las tareas recibidas. Se listan los bloqueos existentes que impiden continuar al equipo su labor si los hubiera (*Impediments Backlog*). El equipo desarrolla y evoluciona el producto con todas las tareas (de esta forma vemos que salvo en el *Sprint 0* se produce una nueva versión ejecutable).

Sprint review: se realiza una demo al *Product Owner*. Se analiza el *Sprint* y se comienza el *Improvement Period* en el que se realizan mejoras sobre el proyecto antes de comenzar una nueva iteración.

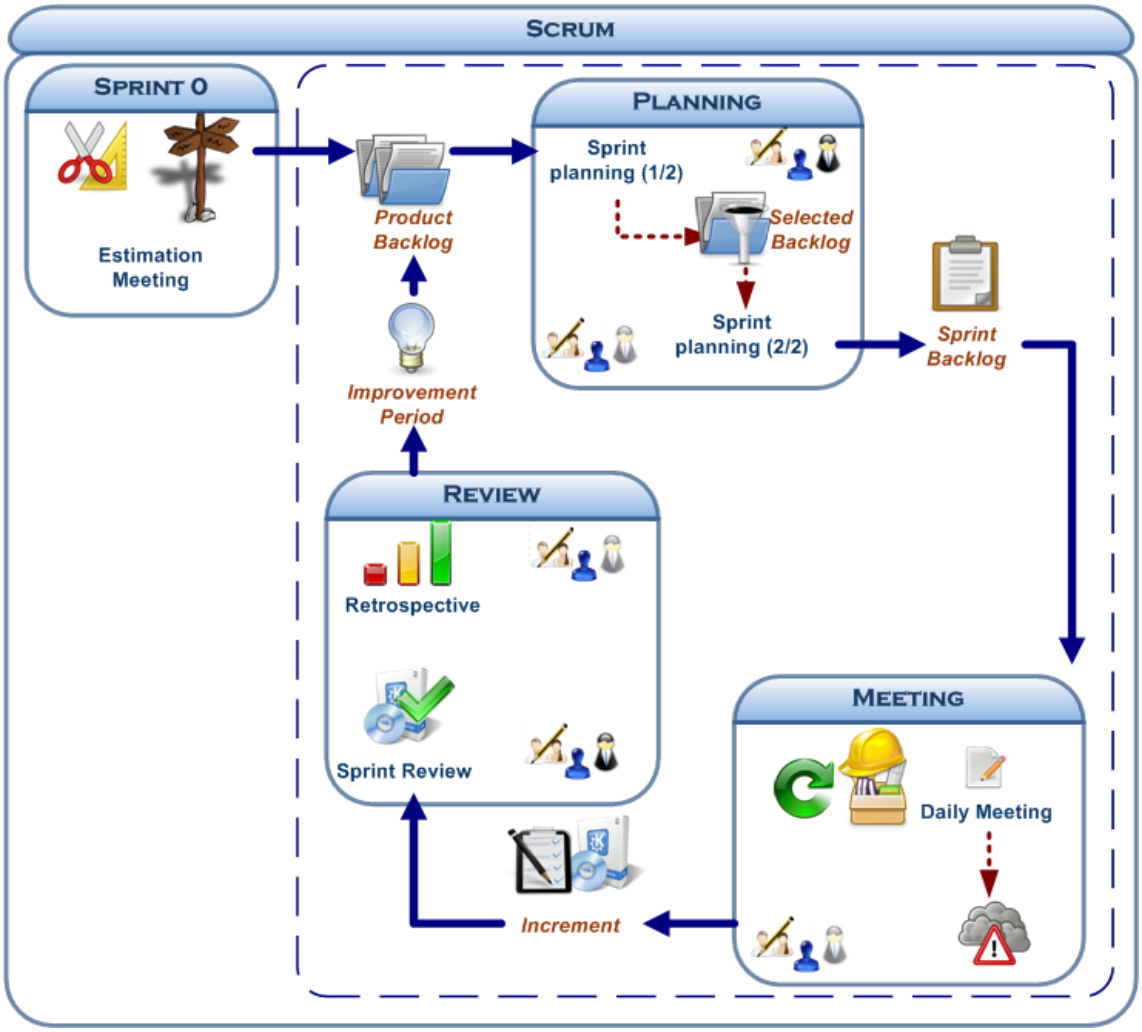


Ilustración 5 Metodología Scrum

Acorde con esta metodología hemos seguido una serie de *sprints* o hitos para el desarrollo de nuestro prototipo. En la siguiente sección ofrecemos las principales tecnologías empleadas durante estos hitos.

PRINCIPALES TECNOLOGÍAS APLICADAS

En el desarrollo de nuestro prototipo hemos realizado 5 iteraciones como veremos en el apartado *Desarrollo del sistema para la extracción de atributos (Features)*. A continuación ofrecemos el compendio de tecnologías usadas para este desarrollo ya que forman parte de este capítulo que es más teórico. Nos basamos en las iteraciones realizadas para mencionar la introducción de nuevas tecnologías usadas a destacar:

Hito 1: Java, XML, DOM, XPath, log4j y CVS.

> *Java* (35)*:* es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. Este lenguaje tiene mucha similitud con la sintaxis de C y C++ pero simplifica el modelo de objetos y elimina herramientas de bajo nivel por la tendencia a producir errores en su

manipulación. Cabe destacar que corre sobre una máquina virtual de manera que es independiente de la plataforma.

XML (36)*:* del inglés, *Extensible Markup Language*. Es un metalenguaje extensible de etiquetas desarrollados por el *World Wide Web Consortium* (W3C). Nació como una simplificación y adaptación del *Standard Generalized Markup Language* (SGML) y permite definir lenguajes específicos. Es un estándar de facto para el intercambio de información estructurada entre diferentes plataformas.

DOM (37)*:* del inglés, *Document Object Model. E*s una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos *Hypertext Markup Language* (HTML) y XML, de esta forma es posible acceder y manipular los datos representados. Se puede acceder y modificar tanto el contenido como la estructura.

XPath (38)*:* del inglés, *XML Path Language*. Es un lenguaje para acceder a partes de un documento XML. Mediante este lenguaje se construyen expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto plano. Mediante XPath se pueden realizar búsquedas y selecciones teniendo en cuenta la estructura jerárquica del XML. Fue diseñado para su uso combinado en el estándar *Extensible Stylesheet Language Transformations* (XSLT) para seleccionar la estructura del documento como parte de la entrada de la transformación.

Log4j (39)*:* se corresponde con una biblioteca open source desarrollada en Java por la fundación Apache Software y permite elegir la salida y el nivel de granularidad de los mensajes en tiempo de ejecución. Esta configuración se realiza en tiempo de ejecución ya sea mediante código o mediante ficheros de configuración.

CVS (40)*:* del inglés, *Concurrent Versions System*. Es una utilidad informática que implementa un sistema de control de versiones, manteniendo el registro del trabajo realizado en un proyecto por los desarrolladores junto con los cambios que éstos han realizado a lo largo del tiempo. Esto permite la colaboración entre desarrolladores en el desarrollo de proyectos.

Hito 2: XSD, Castor, OpenKapow, Servicios REST, Apache Commons HTTP Client y CSV.

XSD (41): *XML Schema* (XSD) es un lenguaje desarrollado por el W3C que se usa para definir la estructura y las restricciones en cuanto a contenido y tipo en documentos XML de forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. De esta forma, se puede tener una abstracción de alto nivel correspondiente al documento.

Castor (42): es un *framework open source* desarrollado en Java para enlazar datos entre objetos Java, documentos XML y bases de datos relacionales. Castor posibilita tanto el enlace de Java con XML, persistencia de Java con *Structured Language Query* (SQL) y otros más.

OpenKapow (43)*:* es un potente extractor de información de cualquier sitio web convirtiéndola en servicios *Representational State Transfer* (REST) y en *Rich Site Summary (*RSS) *feeds*. De manera que se puede tener acceso estructurado a datos que proceden de cualquier fuente web. Para ello el usuario define un robot mediante una interfaz visual que es el encargado de recabar la información, ese robot se sube a un servidor web y se accede a los datos ofrecidos mediante protocolo *Hypertext Transfer Protocol* (HTTP). Desde diciembre del 2009 el servidor ha dejado de ser de libre uso y ha pasado a denominarse StrikeIron, en el que se ofrecen los servicios mediante el modelo de negocio *Software as a Service* (SaaS) cobrando por la demanda.

Servicios REST (44)*:* La transferencia de Estado Representacional es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web (WWW). Si bien, actualmente se usa en el sentido más amplio para definir las interfaces que utilizan XML y HTTP sin abstracciones adicionales como las usadas por otros protocolos de intercambio de mensajes como *Simple Object Access Protocol* (SOAP).

Apache Commons HTTP Client (45)*:* Dado que el paquete de java java.net ofrece una funcionalidad muy básica de acceso a recursos vía HTTP, este componente implementa los últimas recomendaciones y estándares relacionados con HTTP en el lado del cliente. De esta forma se aporta un componente sencillo, con múltiples posibilidades y robusto frente al estándar de HTTP. Está orientado para aplicaciones cliente como navegadores web, clientes de servicios web y sistemas que extienden o usan el protocolo HTTP en sistemas distribuidos.

CSV (46)*:* es un formato abierto y sencillo que se utiliza para representar datos en forma de tabla, en la que las columnas se separan por comas y las filas por saltos de línea. Los campos que contengan una coma, salto de línea o una comilla doble deben delimitarse mediante comillas dobles. No lleva asociado ni juego de caracteres ni el formato para saltos de línea, de manera que todo esto se debe indicar al abrir el fichero.

Hito 3: Lucene y Snowball.

Lucene (47)*:* es un *Application Programming Interface* (API) de código abierto que sirve para recuperar información. Su licencia es Apache Software License. Es útil

en la implementación de motores de búsqueda por sus propiedades de indexación y búsqueda en texto completo

Snowball (48)*:* es un lenguaje diseñado para procesar cadenas de caracteres y crear algoritmos de *stemming*, es decir, implementar analizadores de palabras para extraer sus raíces. Cubre múltiples idiomas y existen implementaciones en distintos lenguajes de programación. Lucene hace uso de las capacidades de *stemming* de Snowball para poder realizar búsquedas.

Hito 4: Web-Harvest, XQuery, XHTML.

Web-harvest (49)*:* es una herramienta de extracción de datos de la Web. Mediante varias tecnologías – XSLT, *XML Query Language* (XQuery), XPath, expresiones regulares etc. – se puede manipular XML. Principalmente se centra en el trabajo con HTML y XML procedente de páginas web o RSS.

XQuery (50)*:* es un lenguaje de consulta diseñado para consultar colecciones de datos XML. Su semántica se asemeja a SQL, pero incluye algunas capacidades de programación. Proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML, como por ejemplo bases de datos relacionales o documentos ofimáticos.

XHTML (51)*: eXtensible Hypertext Markup Language* (XHTML) es un lenguaje de marcado diseñado para sustituir al HTML cumpliendo con todas las funcionalidades de HTML pero adoptando las especificaciones de XML. Aporta valor en el avance del W3C hacia una web semántica donde la información y la forma de representarla estén separadas.

Hito 5: Ninguna tecnología nueva aportada.

Capítulo 3. SVM aplicado a la extracción de interacciones farmacológicas en textos biomédicos

En este capítulo analizamos en profundidad nuestro sistema de extracción de IFs. Para ello nos centramos en conocer la base del sistema: el corpus DrugDDI, el cual nos proporciona los datos básicos que formarán parte de la extracción en nuestro sistema.

Construimos nuestro sistema de extracción de IFs generando un fichero con características provenientes del corpus DrugDDI (este corpus fue extraído de textos en lenguaje natural). Además extraemos características de recursos externos como son DrugBank y WhoCC para obtener enriquecer el conocimiento sobre los fármacos de nuestro sistema. Posteriormente procesamos el resultado obtenido mediante el algoritmo SMO de manera que, al estar anotado el corpus, analizamos los resultados y comparamos los resultados obtenidos en la tesis (1).

El corpus DrugDDI (5)

En el dominio biomédico existen numerosos corpus anotados para realizar experimentos, la mayoría dedicados a la interacción de proteínas y/o genes (como ejemplos tenemos: Critical Assesment of Information Extraction systems in Biology (BioCreATivE) – proteínas en el hombre -, BioText Data – dataset para la extracción de relaciones entre enfermedades y su tratamiento –, DIPPPI – anotaciones sobre interacciones entre proteínas-, IEPA – contiene anotaciones sobre interacciones entre proteínas sobre resúmenes de Medline -, MedTag – anotados nombres de genes y proteínas -, TREC Genomics – para anotación de genes - , Yapex –datos para entrenamiento y pruebas para el etiquetador de Yapex. Sin embargo, hay una gran carencia en el ámbito de la interacción entre medicamentos. Como base para nuestro trabajo usaremos el corpus DrugDDI (5) que es el único extraído mediante técnicas de ingeniería lingüística para este tipo de relaciones. Este corpus, aparte de estar anotado, dispone de documentación que nos facilita el trabajo y es libre para uso académico. Este corpus pretende asentarse como un estándar de facto para la evaluación y aplicación de técnicas que extraigan interacciones farmacológicas.

Es por esto que indagaremos la procedencia de este recurso para poder conocer mejor las capacidades que nos puede brindar. A modo de resumen podemos indicar que para componer este corpus, su autora Isabel Segura (1), recurre a fuentes externas que aportan el conocimiento biomédico: DrugBank. DrugBank se usa para extraer los textos de interacciones entre fármacos que se van a anotar y también como medio de conexión con otros recursos biomédicos que enriquecen el sistema. El corpus DrugDDI nace a

partir de la extracción de estos textos. A continuación se integra en el sistema el uso de la herramienta UMLS Metamap Transfer (MMTx), la cual sirve para realizar el análisis sintáctico y semántico de los documentos del corpus. De esta forma se realiza un mapeo entre el texto analizado y los conceptos de UMLS. La herramienta MMTx es la encargada de dividir en oraciones los textos, tokenizar, realizar el POST, el análisis sintáctico y enlazar las frases con conceptos UMLS. Finalmente se desarrolla y utiliza la herramienta visual: DDIAnnotate tool para anotar las interacciones. Esta herramienta es capaz de resaltar los elementos correspondientes a fármacos y añadir la interacción correspondiente entre cada par de fármacos.

DrugBank

DrugBank (52) es una base de datos que representa y enlaza gran cantidad de información sobre fármacos. Combina datos químicos, farmacológicos y farmacéuticos. Datos sobre unas 4800 sustancias farmacológicas, enriquecidos con datos de más de 3000 fármacos experimentales. Entre la información que aporta nos encontramos con datos como la nomenclatura, estructura, propiedades físicas de los medicamentos, marcas de medicamentos, fórmulas químicas, dosis, toxicidad, además de para qué dolencias están recomendadas. Con conexiones con otras bases de conocimiento como pathway[1] donde se describen los procesos relacionados con el fármaco en distintos organismos, también conecta con Who Collaborating Center for Drug Statistics Methodology [2] de manera que enlaza con las familias a las que pertenecen los códigos *Anatomical Therapeutic Chemical* (ATC) asociados al fármaco, con PubMed[3] que es un servicio americano que ofrece gran cantidad de información sobre revistas científicas de medicina y la DBPedia[4] que es una gran enciclopedia mantenida por la comunidad con información estructurada, entre otros. Toda esta información está destinada al uso por los profesionales sanitarios y del dominio biomédico por su carácter técnico. Algunas ventajas de esta fuente de información es su disponibilidad, ya que la encontramos accesible online, es libre y está mantenida por la universidad de Alberta.

Cuenta también con una colección de datos sobre IFF para los fármacos. Podemos ver los textos extraídos de diferentes fuentes médicas en el campo *Interactions* de las fichas de los fármacos. Estos datos son verificados por farmacéuticos acreditados y añadidos manualmente a la base de conocimiento. También disponemos del campo *Drug Interactions* en el que aparecen de forma estructurada interacciones con otros

[1] http://pathman.smpdb.ca/

[2] http://www.whocc.no

[3] http://www.ncbi.nlm.nih.gov/pubmed/

[4] http://dbpedia.org

fármacos. Precisamente la información correspondiente al campo *Interactions* es tomada como la fuente de datos a procesar en el corpus DrugDDI ya que se considera que es de confianza y representativa como forma de expresar comúnmente las IFF. Estos textos son muy similares a los que nos podemos encontrar dentro de los resúmenes de Medline que están orientados al ámbito farmacéutico.

Debido a que el trabajo de anotación es muy costoso, se selecciona una porción de estos textos para realizar el procesado y la posterior anotación.

> *Potassium-depleting diuretics are a major contributing factor to digitalis toxicity. Calcium, particularly if administered rapidly by the intravenous route, may produce serious arrhythmias in digitalized patients. Quinidine, verapamil, amiodarone, propafenone, indomethacin, itraconazole, alprazolam, and spironolactone raise the serum digoxin concentration due to a reduction in clearance and/or in volume of distribution of the drug, with the implication that digitalis intoxication may result. Erythromycin and clarithromycin (and possibly other macrolide antibiotics) and tetracycline may increase digoxin absorption in patients who inactivate digoxin by bacterial metabolism in the lower intestine, so that digitalis intoxication may result […]*

Ilustración 6 Ejemplo de textos sobre interacciones para la Digoxina (DrugBank)

UMLS

Para el procesado del corpus, el sistema se apoya en *Unified Medical Language Systems* (UMLS) (53) (54). UMLS sirve para poder aportar una descripción precisa dentro de la etapa de procesamiento de los textos y es un conjunto de recursos desarrollados por el *National Library of Medicine* (NLM) con el objetivo de ayudar en el desarrollo de tecnologías sobre lenguaje natural en los ámbitos biomédicos y de salud. UMLS se compone de la integración de tres fuentes de conocimiento: el Metatesauro, la Red Semántica y el Lexicón Especializado.

El Metatesuaro: es el núcleo de UMLS, es una ontología sobre el dominio biomédico que integra numerosas terminologías médicas. Consta de una colección de conceptos y términos extraídos de diferentes vocabularios controlados, incluyendo también sus relaciones. Entre las terminologías que integra nos encontramos con *Medical Subject Heading* (MeSH) (55), *Systematized Nomenclature of Medicine-Clinical Terms* (SNOMED CT) (56) y otras muchas. Además, gracias a su modularidad permite la inclusión de nuevos dominios.

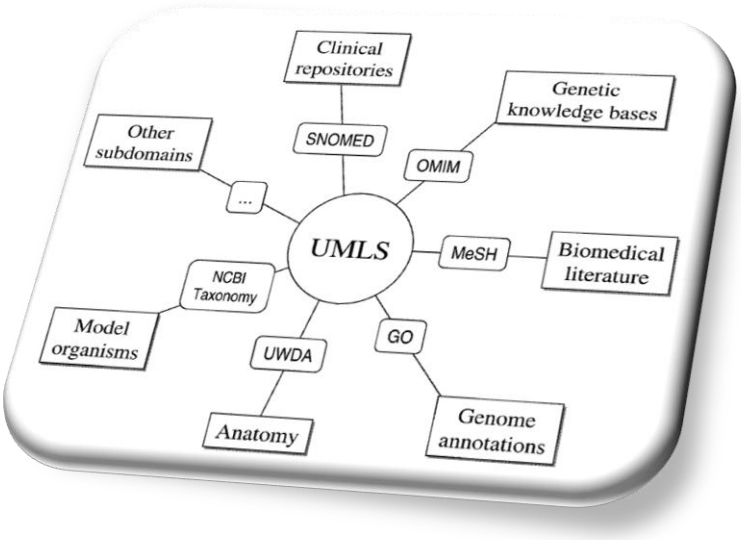


Ilustración 7 Varios subdominios integrados en UMLS

La Red Semántica: contiene un conjunto de categorías y relaciones con la finalidad de poder clasificar y relacionar las entradas del Metatesauro. Esta red consta de 135 tipos semánticos, a modo de ejemplo podemos mencionar algunos de ellos: *Pharmaceutical subtance* (phsu), *Amino Acid, Peptide, or Protein* (aapp), *Disease or Syndrome* (dsyn) o *Gene or Genome* (gngm).

El Lexicón Especializado: es una base de datos con información léxico-gráfica (sintáctica, morfológica y ortográfica) para el uso de Procesamiento de Lenguaje Natural (PLN).

Los conceptos representados en el Metatesauro se categorizan semánticamente mediante las categorías de la Red Semántica. Esto permite conocer la categoría semántica para cada concepto. Además, UMLS no depende de la lengua, de manera que un término puede estar asociado con uno o más conceptos. A cada término le corresponde un identificador único de forma que se relaciona con uno o varios tipos semánticos de las categorías de la Red Semántica. Y también podemos acceder a los términos asociados a un concepto dado éste en cada una de las lenguas tratadas para ese recurso.

Construcción del corpus

Se integra en el sistema el uso de la herramienta UMLS Metamap Transfer (MMTx) (57), la cual sirve para realizar el análisis sintáctico y semántico de los documentos del corpus. De esta forma se realiza un mapeo entre el texto analizado y los conceptos del Metatesauro de UMLS. La herramienta MMTx se encarga del proceso de las oraciones de los textos enlazando las frases con conceptos UMLS. De esta forma se etiquetan todos los datos necesarios y se almacena toda la información transformando la salida

del MMTx a XML y separando las anotaciones del texto mediante *standoff annotation* (58) ya que esto facilita que se puedan recuperar de forma inmediata los textos sin necesidad de accederlos mediante las anotaciones.

Una vez realizado el análisis sintáctico, el MMTx busca las expresiones dentro del Metatesauro, de manera que genera una serie de variantes usando el lexicón y técnicas lingüísticas. Finalmente se asigna un identificador único a cada concepto, denominado *concept unique identifier* (CUI). A su vez se reflejan tanto el nombre del concepto como sus categorías semánticas.

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE SENTENCES SYSTEM "DTD.dtd">
<SENTENCES>
    [...]
    <SENTENCE ID="s3" TEXT="Based on adult data, lower doses of caffeine may
be needed following coadministration of drugs which are reported to decrease
caffeine elimination (e.g., cimetidine and ketoconazole) and higher caffeine
doses may be needed following coadministration of drugs that increase caffeine
elimination (e.g., phenobarbital and phenytoin).">
        <PHRASES>
            [...]
            <PHRASE ID="s3.p47" NUMTOKENS="2" TEXT="caffeine elimination"
TYPE="NP">
                <MAPPINGS>
                    <MULTIMAPCUI>
                        <MAP CUI="C0006644" NAME="Caffeine"
NAME_SHORT="Caffeine" PROB="694" SEMTYPES="orch,phsu" USAN="NO"/>
                        <MAP CUI="C0518891" NAME="Elimination outcomes"
NAME_SHORT="Elimination" PROB="861" SEMTYPES="inpr"/>
                    </MULTIMAPCUI>
                    <MULTIMAPCUI>
                        <MAP CUI="C0006644" NAME="Caffeine"
NAME_SHORT="Caffeine" PROB="694" SEMTYPES="orch,phsu" USAN="NO"/>
                        <MAP CUI="C0221102" NAME="Excretory function"
NAME_SHORT="Elimination" PROB="861" SEMTYPES="phsf"/>
                    </MULTIMAPCUI>
                </MAPPINGS>
                <TOKENS>
                    <TOKEN ISHEAD="false" ORD="0" POS="noun" WORD="caffeine"/>
                    <TOKEN ISHEAD="true" ORD="1" POS="noun"
WORD="elimination"/>
                </TOKENS>
            </PHRASE>
            [...]
        </PHRASES>
        <DDIS>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p35" DRUG_2="s3.p50" ID="1"
NAME_DRUG_1="of caffeine" NAME_DRUG_2="cimetidine" SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p35" DRUG_2="s3.p52" ID="2"
NAME_DRUG_1="of caffeine" NAME_DRUG_2="ketoconazole" SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p55" DRUG_2="s3.p67" ID="3"
NAME_DRUG_1="higher caffeine doses" NAME_DRUG_2="phenobarbital"
SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p55" DRUG_2="s3.p69" ID="4"
NAME_DRUG_1="higher caffeine doses" NAME_DRUG_2="phenytoin"
SEVERITY="undefined"/>
        </DDIS>
    </SENTENCE>
    [...]

</SENTENCES>
```

Ilustración 8 Ejemplo de XML para la cafeína (Corpus DrugDDI)

Tras el análisis de la Red Semántica se identifican como categorías válidas que denoten fármacos las siguientes: *Clinical Drug (cnld), Pharmacological Substance (phsu)* y *Antibiotic (antb)*. También se clasifican entidades abstractas que se corresponden con las clases abstract *pharmacological substances* o *pharmaceutical preparations*. Así la palabra medicamento o fármaco serán catalogadas dentro de estos grupos. Estas anotaciones pueden servir para detectar la anáfora en referencia a un fármaco nombrado anteriormente con lo que no se pierden las interacciones que implican estas referencias.

Para terminar, debemos mencionar que el corpus DrugDDI contiene un total de 3.160 interacciones farmacológicas identificadas. Además, estas anotaciones de las interacciones se han realizado a nivel de oraciones y han sido contrastadas gracias a la ayuda de un investigador con experiencia en farmacología y un farmacéutico experimentado. El conjunto de datos anotados corresponde a 3.774 frases que contienen al menos dos fármacos, de las cuales, sólo 2.043 contienen al menos una interacción etiquetadas con nodos *DDI* dentro del XML.

Desarrollo del sistema para la extracción de atributos (Features)

Para poder realizar un estudio de clasificación mediante aprendizaje automático, se deben tener en cuenta las necesidades que ello conlleva. Por un lado deberemos tener en cuenta la extracción de atributos que será necesaria para alimentar al clasificador y por otro lado deberemos tener en cuenta que la salida del sistema debe ser compatible con los formatos aceptados por el clasificador.

Para este desarrollo emplearemos metodologías de desarrollo ágil para la ejecución de la implementación marcando unos hitos base en los que se obtienen prototipos funcionales que evolucionan las características del anterior. Entre otros elementos tenemos en cuenta que la definición del corpus es cambiante ya que al inicio está aún en fase de desarrollo. Además los requisitos y necesidades van aumentando de manera que el sistema debe evolucionar para adaptarse a las nuevas necesidades rápidamente.

Hitos con las evoluciones marcadas por los sprints dentro del proyecto

Siguiendo con la metodología Scrum, definimos los hitos producidos durante la evolución del sistema que han dado lugar a distintos prototipos funcionales:

Hito 1: este hito nace por la necesidad de asentar la base del sistema propiciando un sistema de salida para ejemplos positivos y negativos. Es pues la base del prototipo, y en él se considera la lectura de los ficheros del corpus DrugDDI obteniendo exclusivamente

las frases con únicamente dos fármacos por simplificación, recabando los datos correspondientes al nombre de los fármacos. Además se realiza la implementación de logs para la posible detección de errores del sistema y la generación de un fichero básico de salida con los ejemplos obtenidos.

Hito 2: se desarrolla este hito bajo la premisa de adaptarse a la evolución del corpus DrugDDI mediante mejoras en el sistema de extracción de datos con la inclusión de nuevas tecnologías. Además surge la necesidad de adquirir conocimiento de fuentes externas. Por estos motivos se desarrolla la evolución del prototipo con un sistema de extracción basado en Castor y XSD sobre el corpus, de manera que se produzca una rápida adaptación en la extracción de datos del corpus. Se procede también a la implementación de sendos robots sobre la tecnología de OpenKapow para la extracción de códigos ATC de la base de conocimiento de *DrugBank* y de familias de la base de conocimiento de *WHO Collaborating Centre for Drug Statistics Methodology (WHOCC)*. De momento, por simplificación y rápida evolución, en caso de que el fármaco esté asociado a más de un código ATC, se escoge el primero. También se modifica el módulo de generación del fichero de salida, con la implementación de la generación de fichero CSV como salida compatible con Weka dado que en el anterior hito no se había contemplado esta necesidad.

Hito 3: se aplica una nueva evolución del prototipo extrayendo más información que se considera de relevancia, adaptándonos a las necesidades del algoritmo SMO, mejorando los robots para extraer todas las familias ATC de un fármaco y mejorando la eficiencia ya que las llamadas a recursos externos consumen mucho tiempo. Así se añade la detección de verbos de interacción, negaciones y anotaciones DDI dentro del corpus (estas anotaciones darán la clave en cuanto a ejemplos positivos y negativos). Al modificar los robots para extraer todas las familias relacionadas a un fármaco también se modifica la salida del fichero CSV para determinar las familias mediante vectores de valores booleanos, es decir, la representación pasa de ser del tipo nominal a un vector de valores binarios que representan todas las familias existentes (esto nace como una necesidad dentro de SVM como vimos en el apartado SMO), marcando con 0 en caso de no pertenecer a la familia y con 1 en caso de pertenecer a la familia. Con esto último se contemplan los fármacos que pertenecen a más de una familia. Para recoger todas las familias se genera un nuevo robot sobre WHOCC de manera que recorre y almacena todas las familias existentes. Para mejorar la eficiencia y disminuir el número de consultas al exterior se implementa un sistema de caché que almacena los resultados de llamadas anteriores, así que si existe la información necesaria en esta caché ya no se realizará la ejecución del robot.

Hito 4 (de contingencia): ante la conversión de los servicios de OpenKapow en servicios de pago, se realiza una iteración para cambiar el módulo basado en OpenKapow por un módulo que cumpla con la misma misión (extraer los atributos de *DrugBank* y *WHOCC).*

El módulo cambiado se basa en tecnología open source con licencia BSD: la tecnología elegida es Web-Harvest. Se produce la migración tecnológica a esta nueva tecnología que también produce un aumento de eficiencia al ejecutarse en modo local y no sobre un servidor externo como se hacía en el módulo anterior.

Hito 5: se aplica esta evolución para agregar ejemplos negativos ya que hasta el momento sólo estábamos considerando los positivos, también se mejora el algoritmo de extracción para contemplar las oraciones con más de dos fármacos. Realizamos el cambio en la forma de tener en cuenta la negación y los verbos de interacción, sólo se considera su existencia en el caso de que se encuentren entre dos fármacos marcados como con interacción (anotados en el DDI). Inclusión en el fichero CSV de la presencia de verbos modales teniendo en cuenta que su presencia sea entre dos fármacos marcados con interacción. También se produce en este hito la modificación del algoritmo de extracción de atributos del corpus DrugDDI para tener en cuenta todas las frases aunque tengan más de dos fármacos que interaccionen, incluyendo también los ejemplos negativos.

Arquitectura del sistema

Hemos llamado a nuestro sistema Druida (DRUg Interaction Detectection and Assesment). Este sistema procesa los ficheros del corpus y encapsula el algoritmo de extracción de datos y generación de las instancias de ejemplos de IFs encontradas. Finalmente se procesan los ejemplos mediante aprendizaje automático con el uso de SMO.

Podemos distinguir varios componentes en nuestro sistema:

Recursos locales y externos: contamos con una serie de recursos como son el corpus DrugDDI (local), DrugBank y WhoCC (externos) que nos sirven para extraer características.

Recoletor: es el encargado de acceder a los recursos locales (XML) para recabar la información necesaria de forma que se puedan generar parte de las características que compondrán el fichero de ejemplos de IFs. Se extraen características relacionadas con los fármacos y características relacionadas con el contexto que nos encontramos en la oración que contiene las interacciones.

Stemmer: este elemento se encarga de procesar algunos elementos extraídos por el Recolector para hallar raíces de palabras y poder extraer como características si existen determinadas características o no como puedan ser verbos de interacción o negaciones.

Arañas Web: este componente es el encargado de extraer datos de recursos externos: DrugBank y WhoCC. Cuenta con un sistema de caché que persiste los datos extraídos con el fin de optimizar el rendimiento y minimizar el acceso al exterior.

Atributos IF: embebe la capacidad de construir el fichero CSV de salida que contiene los atributos e instancias generadas y procesadas.

Clasificador: una vez generado el fichero de salida se procesa mediante el Clasificador, que es el que implementa los algoritmos de aprendizaje automático que necesitaremos para procesar los datos y tratar de inferir conocimiento en cuestión de interacción entre fármacos.

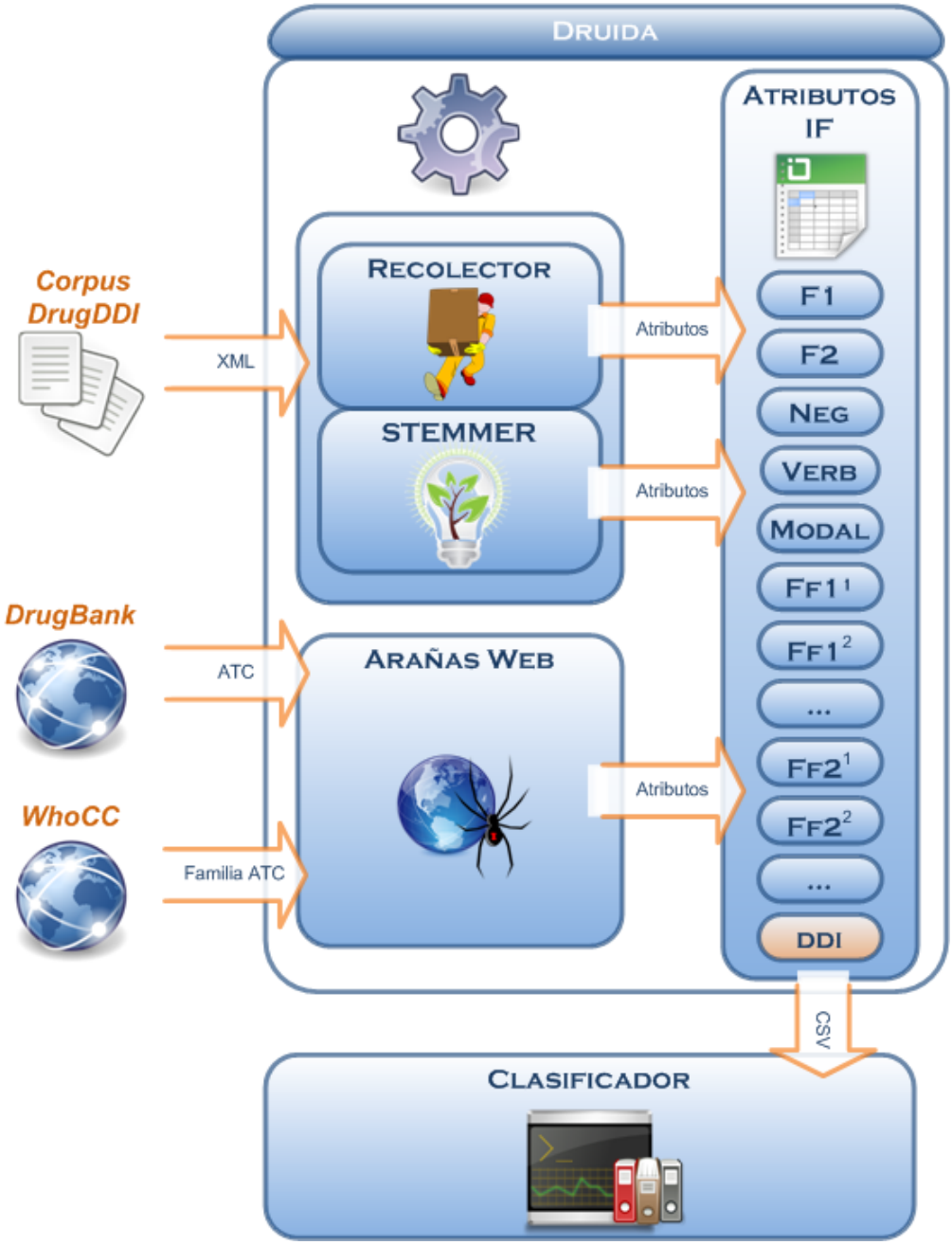


Ilustración 9 Arquitectura del sistema

EXTRACCIÓN DE FEATURES

La extracción de features consta de dos fuentes de información: por un lado el corpus DrugDDI que pertenece a una fuente local y por otro lado las fuentes externas que se corresponden con DrugBank y WHOCC.

Para la recolección de los datos del corpus DrugDDI, en el hito 1 se planteó simplemente el acceso a los datos locales y extracción mediante XPath ya que las primeras necesidades marcadas en este hito eran simplemente acceder dentro de los XML a cada oración y extraer los nombres de los fármacos que contenían. Dado que en el siguiente hito (hito 2) se pretendían extraer más datos y la estructura del XML de los ficheros que conformaban el corpus se vio alterada por su evolución, se decidió realizar una mejora en la extracción de estos datos mediante el empleo de un framework que ofreciera la relación directa entre los datos del XML y objetos java; de igual manera que otras herramientas como Hibernate que ofrecen un mapeo directo entre las relaciones de una base de datos y objetos java que abstraen ese conocimiento y que se denominan *Object-Relation Mapping* (ORM). Para ello optamos por Castor, que a priori es una herramienta muy sencilla que no necesita ningún tipo de configuración previa frente a otros sistemas como XMLBeans, y sin embargo aporta la posibilidad de realizar configuraciones más avanzadas en situaciones más complejas. Para poder realizar la integración de Castor en nuestro sistema como método de acceso a la estructura XML, desarrollamos el esquema XSD correspondiente al XML de los ficheros del corpus. Y mediante un proceso automatizado, generamos los objetos necesarios para tener acceso directo a los datos desde java y poder acceder al contenido.

Una vez integrado Castor en el sistema, el acceso al contenido se simplifica enormemente y facilita el entendimiento del algoritmo de extracción de datos al poseer la misma nomenclatura que el fichero del Corpus. A partir de este momento, mediante la evolución en los siguientes hitos se accede a un mayor número de datos. Dentro del hito 2 se incorporan nuevos elementos en la extracción ya que se accede también a los datos externos mediante minería web de datos. Para ello se construye un robot mediante OpenKapow (Robomaker) el cuál accede a la web con la base de conocimiento de DrugBank, realiza una búsqueda para el fármaco a buscar, accede a su ficha dentro de la base de conocimiento, extrae el primer código ATC (por simplificación) y el nombre asociado. Este robot es evolucionado en el siguiente hito (hito 3) de manera que se añade al proceso anterior la recolección de todos los códigos ATC del fármaco y, mediante la búsqueda dentro de WHOCC de cada código ATC, se añade la familia de cuarto nivel correspondiente al medicamento. Para poder añadir todos las familias ATC a las que pertenecen los fármacos, se crea un nuevo robot que extrae la jerarquía de familias de WHOCC, de manera que la representación de las familias de los fármacos pasa a ser un vector de variables binarias en el que si el fármaco pertenece a la familia se representará con un ‘1’ en la posición correspondiente y con ‘0’ en caso contrario.

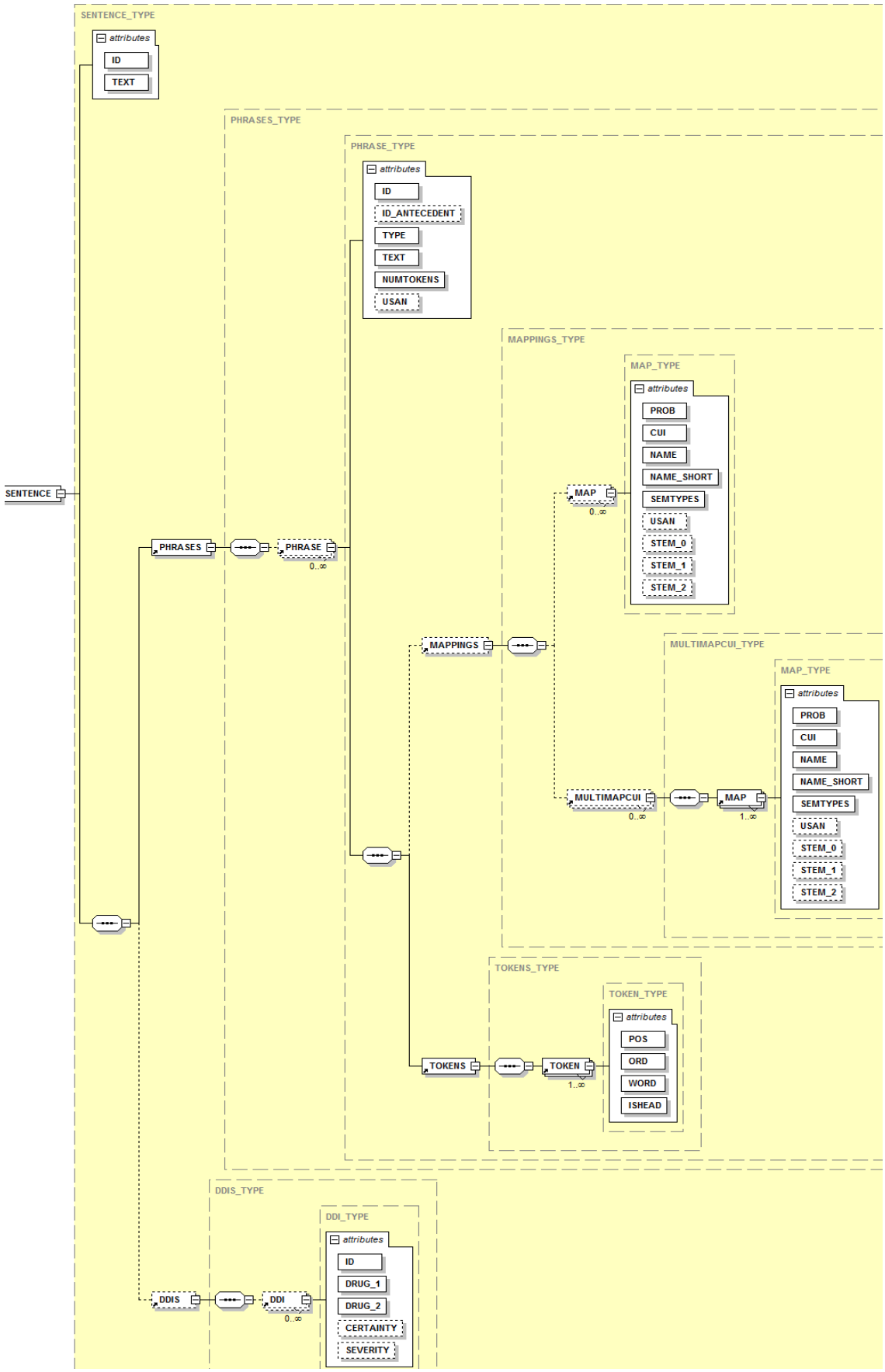


Ilustración 10 Representación gráfica del XSD generado para SENTENCE

También en el hito 3 empleamos Lucene con Snowball para el análisis mediante *stemming* de otros atributos como los verbos de interacción y negación. Por ejemplo, en el caso de los verbos de interacción, se parte de una serie de verbos definidos por un experto del dominio como verbos de interacción, analizamos las raíces de los verbos que nos encontramos en el corpus para comprobar su correspondencia con un verbo de interacción y se especifica así en el resultado.

Debido a la conversión de los servicios de OpenKapow en servicios de pago, se genera en el hito 4 la contemplación de cambio tecnológico a WebHarvest. Este proceso es más costoso debido a que dentro de OpenKapow, su interfaz gráfica y de gran usabilidad permitía diseñar robots robustos en poco tiempo. La adopción de WebHarvest se tomó por su encaje a modo de sustitución del componente de minería web (ambos generan un XML que el sistema abstrae en objetos java entendibles por Druida). También jugaron un papel importante otras características de WebHarvest como que es independiente, open source, con licencia libre, admite acceso a bases de datos, permite persistir la información vía ftp, es extensible y se ejecuta en modo local (OpenKapow ejecutaba el robot de configuración generado sobre un servidor externo).

WebHarvest también se basa en un fichero de configuración en el que se pueden definir las operaciones a realizar en las extracciones. Se basa en tecnologías y estándares libres que sirven para manipular la información recabada de la Web mediante procesadores que realizan distintas tareas: recoger una página web, convertirla en XHTML, tratamiento mediante XPath, XQuery y expresiones regulares (esto dota de la máxima expresividad en las consultas y de las máximas posibilidades en la transformación de la información). Además, en esta última versión, la arquitectura permite la extensión y creación de nuevos procesadores mediante plugins. La interfaz de usuario es parca: un editor XML, aunque en la última versión se ha introducido un depurador básico. Sería maravilloso que se mejoraran los logs de salida en caso de errores, sobretodo en el tratamiento de XQuery ya que la ayuda ofrecida para localizar problemas es muy escasa. A continuación se puede ver una de las funciones implementadas en el sistema para recoger los códigos ATC de un fármaco:

```
<!—
  get ATC Codes from a drug card.
    @param drug: Drug name to search the drug card in www.drugbank.ca
    @return nodes as a list of all ATC codes
 -->
<function name="getATCCodes">
 <return>
  <empty>

   <!-- Search the drug name in drugbank and set the partial URL in the url variable -->
   <var-def name="url">
          <xpath expression="(//table//tr//td/a)[1]/@href">
           <html-to-xml>
            <http url="http://www.drugbank.ca/search/search?query=${drug}"/>
           </html-to-xml>
          </xpath>
   </var-def>

   <!-- build the full URL for the drug card -->
   <var-def name="drugCardUrl">
           <template>${sys.fullUrl("http://www.drugbank.ca/", url)}</template>
   </var-def>


          [...]


          <!-- Process the XML to get the ATC codes -->
          <xquery>
           <![CDATA[ <result> ]]>
            <xq-param name="doc" type="node()">
             <var name="drugCard"/>
            </xq-param>
            <xq-expression><![CDATA[
      declare variable $doc as node() external;


            (: Get all atc codes with XQuery and return them as XML nodes :)
            for $atcCode in $doc//tr[td = "ATC Codes"]//a
                  return <atccode>{data($atcCode)}</atccode>

   ]]></xq-expression>
```

Ilustración 11 Ejemplo de función en Web-Harvest

Así aprovechamos y mezclamos las distintas tecnologías ofrecidas en WebHarvest y aprovechamos esta oportunidad para aprender una tecnología nueva: XQuery. Una vez completado este componente, sustituimos el anterior sistema de minería web por él.

Finalmente, incluimos el atributo DDI que nos indica si realmente existe interacción entre ambos fármacos o no ya que se corresponde con la anotación de un experto del dominio sobre el corpus. Mejoramos el algoritmo de extracción de datos para incluir la existencia de verbos modales y se contemplan todas las oraciones con al menos dos fármacos con interacción entre fármacos. Para ello se emparejan todas las posibilidades de interacción de los fármacos que aparecen en la oración, tomando como ejemplos positivos aquellos que aparecen con una anotación DDI.

Así el vector final con todas las características queda como sigue: inclusión de los códigos ATC para ambos fármacos F1 y F2. Inclusión de todas las familias correspondientes a cada fármaco a modo de vector de booleanos con todos los ATC de familias y marcando a las que pertenezcan: FF1 y FF2; es decir, FF1 es un vector en el que las características de este vector se corresponden con todas las familias ATC de nivel 4 marcando con un valor booleano si el fármaco F1 pertenece o no a cada familia ATC. FF2 es otro vector al igual que el anterior pero con los datos correspondientes a las familias asociadas al fármaco F2. También se incluyen la existencia de negación, existencia de verbo modal y verbo de interacción entre ambos fármacos. Y el campo más importante: DDI, que representa la etiquetación de interacción anotada en el corpus entre ambos fármacos. Existen algunos casos especiales donde no se puede recuperar el código ATC del fármaco o no existen datos para las familias en DrugBank; en el caso de que no se pueda recuperar el código ATC del fármaco, ese fármaco en cuestión tendrá un valor vacío (‘’) por defecto y su vector de familias estará formado por valores negativos. En el caso de que no se disponga de información respecto a las familias en DrugBank, el vector de familias correspondiente a dicho fármaco estará representado por vacíos.

Ilustración 12 Vector de características

Como resultado final del tratamiento de los ficheros XML que forman parte del corpus, obtenemos un total de 17.547 ejemplos de 1.550 características.

VOLCADO DE CARACTERÍSTICAS (FEATURES)

Para el volcado de *features* se ha recurrido al tratamiento mediante log4j. Aprovechando las cualidades de manejo de ficheros y recursos E/S que posee log4j, implementamos un *PatternLayout* de salida a nuestra medida para el volcado de datos, de forma que se pueda enriquecer con los nuevos datos a volcar correspondientes a cada nueva iteración. De esta forma enviamos el objeto de salida que contiene todos los datos al *logger* de log4j y éste escribirá mediante nuestro patrón de salida asignado en el fichero correspondiente al CSV todos los datos necesarios en el formato predefinido.

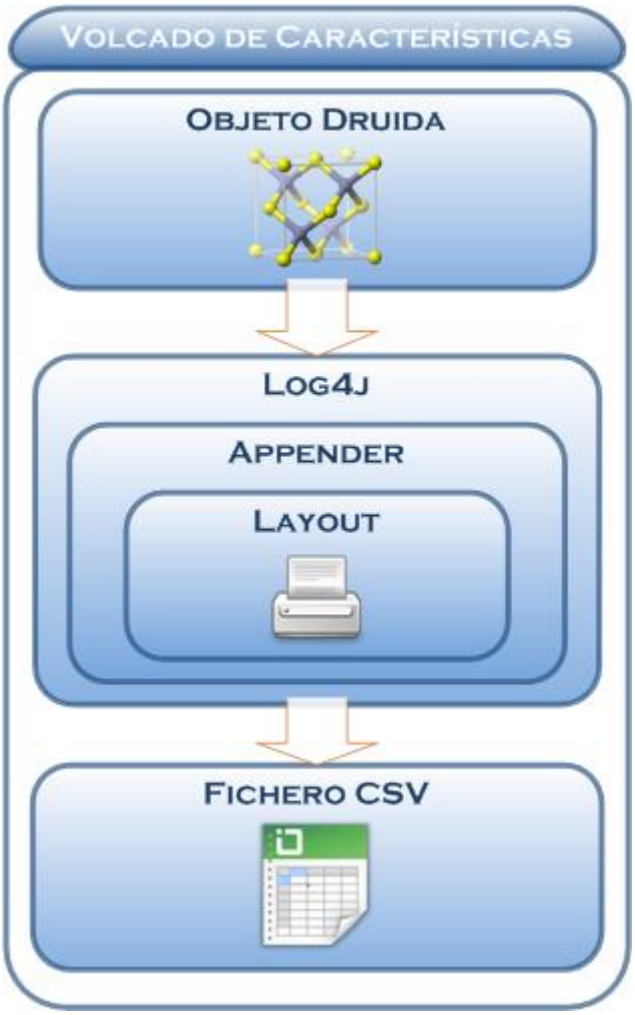


Ilustración 13 Detalle del volcado de características (features)

EXPERIMENTOS

A continuación pasamos a describir los experimentos realizados. Hemos realizado dos experimentos aprovechando los datos de entrada de los que disponemos. Para uno de los experimentos tenemos en cuenta la información asociada a las familias de los fármacos, mientras que para el otro no tendremos en cuenta esta información.

Usamos el algoritmo SVM mediante su implementación SMO como elegimos en el capítulo 2. En Weka, el propio SMO normaliza las características de entrada nominales de acuerdo con sus necesidades ya que no son admitidas estas características directamente. Para nuestros experimentos, con el fin de evitar el problema de coste de tiempo en el entrenamiento del algoritmo, reducimos el número de ejemplos escogiendo los primeros 10.010 ejemplos. Usamos *cross-validation* con 3 *folds* para realizar el entrenamiento y el test de cara a sacar conclusiones sobre los resultados obtenidos.

Experimento sin Códigos ATC de familias

Para este experimento partimos de 10.010 ejemplos con 6 características: fármaco 1, fármaco 2, partícula negativa, verbo modal, verbo de interacción y la anotación de interacción (DDI). Aplicamos SMO y obtenemos los siguientes resultados para la precisión, cobertura y medida-F por cada clase ("DDI": interacción entre fármacos y "No DDI": sin interacción entre fármacos,) así como la macro media:

	Precisión	Cobertura	Medida-F
No DDI	*0,891*	*0,981*	*0,934*
DDI	*0,451*	*0,116*	*0,184*
Macro	**0,671**	**0,548**	**0,559**

Tabla 4 Resultados sin códigos ATC de familias

Como vemos, la macro media tanto para precisión, cobertura y medida-F es alta y nos indicaría que, en media, el sistema está clasificando abarcando gran número de ejemplos y con buena precisión las clases. Sin embargo, si acudimos a los datos individuales por clase nos damos cuenta de que aunque para la clase sin interacción entre los fármacos (No DDI) es cierto que las medidas son muy buenas tanto en precisión, cobertura y medida-F, vemos por otro lado que la clasificación de interacción (DDI) es pésima, con muy mala cobertura (apenas cubre un 11% de esta clase), con un precisión baja de 0,451, y una combinación de ambas mediante la medida-F muy baja. Acudimos a la matriz de confusión para extraer más detalles:

		Resultado correcto	
		DDI	No DDI
Resultado inferido	DDI	139	169
	No DDI	1.060	8.642

Tabla 5 Matriz de confusión sin códigos ATC de familias

Nos damos cuenta de que la proporción entre ejemplos de ambas clases está muy desequilibrada con 88% de ejemplos sin interacción (no DDI), esto influye en que en la macro media los resultados sean buenos, no obstante, ya que nos interesa descubrir interacciones, los resultados son de baja calidad. A pesar de que el algoritmo SMO es robusto ante desequilibrios de conjuntos de datos no balanceados, quizá el excesivo desequilibrio repercuta sobre el entrenamiento y hubiera sido necesario producir un mayor equilibrio, pero nuestra pretensión es ajustarnos a los desequilibrios que se producirían en la extracción de IF en la realidad con conjuntos desbalanceados. Vemos como de los 1.199 ejemplos de IF sólo 139 son clasificadas correctamente: este

resultado no nos sirve para la vida real. Para el caso de inexistencia de IF sólo se fallan 169 de 8.811.

Experimento con Códigos ATC de familias

En este otro experimento, partimos de los mismos ejemplos que en el experimento anterior: los 10.010 ejemplos, pero esta vez con 1.550 características, correspondientes a las del anterior experimento más los vectores que conforman las familias de los fármacos. Aplicamos la misma implementación SMO del algoritmo SVM, obteniendo los siguientes resultados:

	Precisión	Cobertura	Medida-F
No DDI	*0,892*	*0,979*	*0,933*
DDI	*0,447*	*0,127*	*0,198*
Macro	**0,669**	**0,527**	**0,565**

Tabla 6 Resultados con códigos ATC de familias

En este caso los resultados son muy similares al anterior experimento, nos encontramos con el problema de que el sistema es de baja calidad detectando las interacciones entre fármacos, y es muy bueno detectando que no existen interacciones (lo cual es más fácil por el desequilibrio existente). Para la detección de las IFs nos encontramos con que mejora ligeramente la cobertura, con lo que se cubre más volumen de éstas, sin embargo empeora ligeramente la precisión. En las macro medias obtenemos los mismos resultados, y simplemente se produce una ligera mejora en la macro media para la medida-F.

		Resultado correcto	
		DDI	No DDI
Resultado inferido	DDI	152	188
	No DDI	1.047	8.623

Tabla 7 Matriz de confusión con códigos ATC de familias

Como vemos en la matriz de confusión, los ejemplos clasificados correctamente para las IFs (DDI) mejoran sólo en la detección de 13 ejemplos más que en el experimento anterior. Algo que comparado con el conjunto de datos se traduce en una mejora poco perceptible como hemos podido ver en la tabla de precisión, cobertura y medida-F.

Como vemos, la variación entre ambos ejemplos es mínima y no la consideramos válida para realizar una explotación en la vida real. Nos internamos en la siguiente sección en

la discusión entre los resultados obtenidos con funciones *kernels* en la tesis (1) y los obtenidos en este apartado mediante características.

Discusión

En este apartado realizamos la comparación con los resultados obtenidos en los experimentos realizados en este proyecto frente a los mejores resultados obtenidos en la tesis (1).

En la siguiente tabla observamos los resultados obtenidos en los experimentos realizados con SVM basado en características y el mejor experimento basado en *kernels* de la tesis en la detección de IFs (DDI):

	Precisión	Cobertura	Medida-F
Características Sin códigos ATC de Familias	0,451	0,116	0,184
Características Con códigos ATC de Familias	0,447	0,127	0,198
Funciones *kernels* (tesis)	0,57	0,72	0,64

Tabla 8 Resultados métodos basados características y kernels

Como vemos, podemos observar unos mejores resultados en el empleo de funciones *kernels*. Existe una diferencia sustancial entre la precisión obtenida en los experimentos realizados en este proyecto y los obtenidos en la tesis, siendo éstos últimos mejores. El problema aumenta en la cobertura, en el que se denota que la porción de interacciones inferidas sobre el total de interacciones es muy baja (se encuentra entre el 11,6% y el 12,7%). Finalmente, la medida-F, que agrupa a las medidas anteriores, se ve repercutida negativamente por los resultados obtenidos en los experimentos de este proyecto. En la siguiente gráfica podemos ver las diferencias entre los tres experimentos:

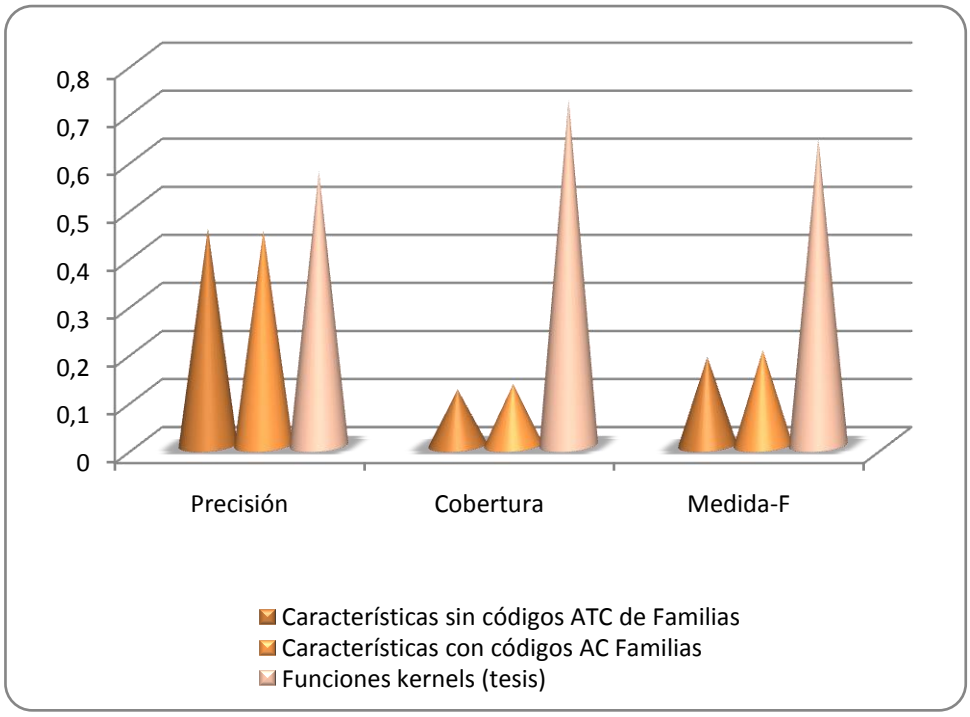


Ilustración 14 Imagen comparativa: características vs kernels

Esto nos indica que no es más efectivo el uso de SVM basado en características frente a funciones *kernels*, al menos con los vectores de características escogidos, pues como hemos visto antes, los resultados para extraer las IFs no son muy buenos. En los experimentos realizados en este proyecto se han obtenido muy buenos resultados en la detección de fármacos entre los que no hay interacción, quizá debido a que los ejemplos positivos y negativos se encuentran muy desequilibrados (muchos ejemplos negativos). En los experimentos realizados en la tesis con conjuntos no balanceados también se ven unos buenos resultados para clasificar fármacos que no tienen IFs, mientras que los resultados empeoran con el intento de detección de fármacos con IFs.

Comprobamos que, como habíamos visto en el estudio de SVM los resultados son muy similares a pesar de modificar las características escogidas en ambos experimentos, con lo que constatamos que SVM es robusto ante la inclusión de características que no son importantes.

Nuestros experimentos también están orientados a la detección de interacciones considerando que los elementos que componen el contexto aportan datos significativos si se encuentran entre ambos fármacos en estudio. Esto quiere decir que sólo consideramos como positivos cuando los elementos que denotan interacción están entre ambos fármacos (verbos de interacción, negación, verbo modal). Debiéramos tener en cuenta también los tres tipos de contexto (*before, between y after*) considerados en la tesis para intentar mejorar los resultados, ya que se pierden los tipos *before* y *after* en los que los elementos se dan antes de ambos fármacos y después respectivamente.

A continuación, en el último capítulo, mencionamos las conclusiones generales sobre este proyecto y en concreto sobre los experimentos realizados. Además se aporta la visión de futuro para poder mejorar este sistema.

Capítulo 4. Conclusiones y trabajos futuros

Gracias a la tesis (1) nos hemos dado cuenta de lo importante que es poder extraer interacciones entre medicamentos para mejorar la seguridad del paciente e incluso salvar vidas; este hecho ha sido la motivación principal de nuestro proyecto. Es muy importante que el personal médico disponga de manera rápida de estos datos, y también toda la población se puede beneficiar de información relativa a las IFs que se pueden producir en su organismo tras la ingesta de varios fármacos.

Hemos comprobado que la aplicación de Castor es muy efectiva en el tratamiento de XML, de manera que facilita el acceso a los contenidos de forma ordenada, intuitiva y comprensible. El empleo de Castor facilita el desarrollo ágil de prototipos ya que simplemente generando un XSD podemos generar de forma automática los objetos necesarios para tratar los datos definidos en un XML; estos objetos recogen la información necesaria del XML. Esto es muy efectivo ante modificaciones del formato del XML, ya que modificando el XSD, sin necesidad de modificar el código manualmente, podemos construir automáticamente los objetos necesarios reduciendo el tiempo de programación. Además, hemos podido comprobar la utilidad de herramientas como OpenKapow y Web-Harvest en la extracción de datos de la Web, que facilitan la extracción y tratamiento de datos generando Servicios Web que facilitan su consumo. Por último hemos visto cómo aplicar de forma inteligente el uso de log4j en el volcado de datos, de manera que cambiar el formato de salida o el entorno de volcado (fichero, base de datos, etc.) sería rápido y poco costoso.

Con nuestros experimentos hemos aportado un granito de arena a la consecución de esta tarea. Aunque los resultados obtenidos no han mejorado los experimentos de la tesis (1), el trabajo desarrollado en este proyecto ha permitido demostrar que el algoritmo SMO y las características utilizadas no son suficientes para detectar interacciones. Tendríamos que investigar más a fondo si otras características aportarían mayor valor en la detección de IFs, y al igual que en los métodos *kernels*, contemplar los tres tipos de contextos posibles (*before*, *between* y *after*) donde se indica que el contexto principal se encuentra antes, entre o después de los fármacos en estudio. Para ello se requeriría una estrecha colaboración con personal especializado que pudiera aportar sus conocimientos del dominio farmacológico con el objetivo mejorar el sistema con la elección de características más adecuadas. Además debiéramos considerar mayor información sintáctica como información del árbol sintáctico más pequeño que incluya ambos fármacos e incluso añadir la información semántica de la que disponemos en UMLS aprovechando el enlace existente del corpus con este sistema. También se podrían realizar nuevos experimentos con otros algoritmos como J48 (59), Naive Bayes (60),etc. con el fin comprobar su efectividad.

Una importante línea de trabajo futuro es la detección de información adicional de las interacciones tales como la severidad, la certeza, las dosis de los fármacos, información temporal, peculiaridades de los pacientes en los que se produce la interaccción. Esta información es vital a la hora de decidir la verdadera significancia clínica de una interacción.

Aunque nos hemos centrado en las IFs como eventos peligrosos, no todas las interacciones son así, también es interesante detectar interacciones entre medicamentos cuyo resultados sea la potenciación del tratamiento (ej. Combinación de antirretrovirales).

Uno de los próximos pasos podría ser la construcción de un sistema dedicado a la detección de interacciones en las publicaciones de biomedicina registradas en MedLine y en otras fuentes de información como PLoS Medicine. Así se podría tener una amplia base de conocimiento que podría ser realimentada si es usada por profesionales de la medicina y enriquecida por esta comunidad. Una posible aplicación directa dentro del dominio médico sería la construcción de un sistema de alertas médicas integradas con el historial clínico de los pacientes, de manera que si a un paciente le administran varios medicamentos, dicho sistema compruebe si existen o no interacciones entre ellos avisando al doctor en caso afirmativo.

Un problema que nos encontramos en las aplicaciones informáticas dedicadas a la medicina es la diferencia entre terminologías referentes a sustancias farmacológicas, diagnósticos, protocolos, etc. Es por ello que el sistema que se construya debiera poseer una normalización de estas terminologías. Para ello, se pueden enlazar los resultados con la terminología más avanzada actualmente: SNOMED CT. Esta terminología ha sido implantada en varios países y sería de utilidad en la atención de pacientes en otros países diferentes al de residencia o en otros hospitales con distintas terminologías.

Finalmente, otra forma de dar un alto empuje a los datos obtenidos sería enriquecerlos mediante su enlace con otros recursos. Esto se puede realizar mediante la triplificación de los datos obtenidos y su conexión con distintos *datasets* de *Linked Open Data* (LOD) (61). Así, se podrían enlazar directamente los fármacos de las interacciones halladas con los ya publicados en el *dataset* de DrugBank y se podrían enlazar los documentos de procedencia de las oraciones tratadas con el proyecto BIO2RDF enlazando con la información disponible de Medline. Existen multitud de fuentes de datos relacionadas con este dominio, así también podemos contar con Linked Life Data (62), ya que agrupa varias fuentes de información como DrugBank, DailyMed, PubMed, UMLS Metathesaurus y UMLS Semantic network entre otros, que podrían servir para enlazar todo tipo de información semántica como relaciones con los conceptos del corpus, los fármacos y los documentos.

Glosario

API *del inglés* Application Programming Interface
ARRF *del inglés* Atribute-Relation File Format
ATC *del inglés* Anatomical Therapeutic Chemical
CSV *del inglés* Comma Separated Values
CUI *del inglés* Concept unique identifier
CVS *del inglés* Concurrent Versions System
DDI *del inglés* Drug-Drug Interaction
DOM *del inglés* Document Object Model
Druida *del inglés* DRug Interaction Detection and Assesment
FN Falsos negativos
FP Falsos positivos
HTML *del inglés* Hypertext Markup Language
HTTP *del inglés* Hypertext Transfer Protocol
IF Interacción farmacológica
IFF Interacción fármaco-fármaco
IPP Interacción proteína-proteína
JDBC *del inglés* Java Database Connectivity
LOD *del inglés* Linked Open Data
MeSH *del inglés* Medical Subject Heading
MMTx *del inglés* MetaMap Transfer
NLM *del inglés* National Library of Medicine
OCP Optimización Cuadrática en subproblemas Pequeños
OCR *del inglés* Optical Character Recognition
ORM *del inglés* Object-Relation Mapping
POST *del inglés* Part-of-speech tagging
RAM Reacción adversa a medicamentos
REST *del inglés* Representational State Transfer
RSS *del inglés* Rich Site Summary
SaaS *del inglés* Software as a Service
SGML *del inglés* Standard Generalized Markup Language
SMO *del inglés* Sequential Minimal Optimization
SNOMED *del inglés* Systematized Nomenclature of Medicine-Clinical Terms
SOAP *del inglés* Simple Object Access Protocol
SQL *del inglés* Structured Language Query
SVM *del inglés* Support Vector Machines
TFN Tasa de Falsos Negativos
TFP Tasa de Falsos Positivos
TVN Tasa de Verdaderos Negativos
TVP Tasa de Verdaderos Positivos
UMLS *del inglés* Unified Medical Language System
VN Verdaderos negativos

VP Verdaderos positivos
W3C *del inglés* World Wide Web Consortium
Weka *del inglés* Waikao Envairoment for Knowledge Analysis
WWW *del inglés* World Wide Web
XHTML *del inglés* eXtensible Hypertext Markup Language
XML *del inglés* Extensible Markup Language
XPath *del inglés* XML Path Language
XQuery *del inglés* XML Query Language
XSD *del inglés* XML Schema Definition
XSLT *del inglés* Stylesheet Language Transformations

Bibliografía

1. **Segura Bedmar, Isabel.** *Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions.* s.l. : Universidad Carlos III de Madrid, 2010.

2. **M, Pirmohamed, S, James and S, Meakin.** Adverse drug reactions as cause of admission to hospital: prospective analysis of 18.820 patients. s.l. : BMJ ; 329: 15-19., 2004.

3. *Adverse drug reactions in patients attending in emergency service.* **Machado-Albal, Jorge E. and Moncada-Escobar, Juan C.** 2, Rev. salud pública [online], Vol. 8, pp. 200-208.

4. *Measuring patient safety in ambulatory care: potential for identifying medical group drug-drug interaction rates using claims data.* **Solberg, L., et al.** 11 Pt 1, 2004, Am J Manag Care, Vol. 10, pp. 753-759.

5. **Segura-Bedmar, Isabel, et al.** DRUGDDI: an annotated corpus for drug-drug interactions. *Bioinformatics.* 2010.

6. Machine Learning. *Wikipedia (The Free Encyclopedia).* [Online] http://en.wikipedia.org/wiki/Weka_%28machine_learning%29.

7. **Witten, I. and Frank, E.** *Data Mining: Practical machine learning tools and techniques.* s.l. : Morgan Kaufmann Pub, 2005.

8. *Weka: Practical machine learning tools and techniques with java implementations.* **Witten, I., et al.** s.l. : Working paper 99/11. Hamilton, New Zealand: University of Waikato, Department of Computer Science, 1999.

9. *Content-based multimedia information retrieval: State of the art and challenges.* **Lew, M., et al.** 1, s.l. : ACM, 2006, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), Vol. 2, pp. 1-19.

10. *Information extraction.* **Sarawagi, S.** 3, 2008, Foundations and Trends in Databases, Vol. 1, pp. 261-377.

11. *Extracting interactions between proteins from the literature.* **Zhou, D. and He, Y.** 2, s.l. : Elsevier, 2008, Journal of biomedical informatics, Vol. 41, pp. 393-407.

12. *Extracting drug-drug interaction articles from MEDLINE to improve the content of drug databases.* **Duda, S., et al.** s.l. : American Medical Informatics Association, 2005.

13. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. [book auth.] G. Erkan, A. Ozgur and D. Radev. *Proceedings of the*

2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Learning (EMNLP-CoNLL). 2007, Vol. 1, pp. 228-237.

14. **Airola, A., et al.** A Graph Kernel for Protein-Protein Interaction Extraction. *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing.* s.l. : Association for Computational Linguistics, 2008, pp. 1-9.

15. *Evaluating contributions of natural language parsers to protein-protein interaction extraction.* **Miyao, Y., et al.** 3, s.l. : Oxford Univ Press, 2009, Bioinformatics, Vol. 25, p. 394.

16. *BioPPISVMExtractor: A protein-protein interaction extractor for biomedical literature using SVM and rich feature sets.* **Yang, Z., Lin, H. and Li, Y.** s.l. : Elsevier, 2009, Journal of Biomedical Informatics.

17. *Supervised Machine Learning: A Review of Classification Techniques.* **Kotsiantis, S.** s.l. : Citeseer, 2007, EDITORIAL BOARDS, PUBLISHING COUNCIL, Vol. 31, pp. 249-268.

18. **Van Rijsbergen, C. J.** Information Retrieval. London : Butterworths, 1979, pp. 112-140.

19. Aprendizaje automático. *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico.

20. *Proper: A Toolbox for Learning from Relational Data with Propositional and Multi-Instance Learners.* **Reutemann, P., Pfahringer, B. and Frank, E.** s.l. : Springer, 2004, AI 2004: Advances in Artificial Intelligence, pp. 1017-1023.

21. *Very simple classification rules perform well on most commonly used datasets.* **Holte, R.** 1, s.l. : Springer, 1993, Machine learning, Vol. 11, pp. 63-90.

22. *Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms.* 2, s.l. : Elsevier, 1992, International Journal of Man-Machine Studies, Vol. 36, pp. 267-287.

23. **Quinlan, J.** *C4. 5: programs for machine learning.* s.l. : Morgan Kaufmann, 1993.

24. *A tutorial on support vector machines for pattern recognition.* **Burges, C.** 2, s.l. : Springer, 1998, Data mining and knowledge discovery, Vol. 2, pp. 121-167.

25. Normalizing Interactor Proteins and Extracting Interaction Protein Pairs using Support Vector Machines. [book auth.] Y. Chen, F. Liu and B. Manderick. *In Proceedings of the BioCreative II. 5 Workshop 2009 on Digital Annotations.* 2009, p. 29.

26. *Mining clinical relationships from patient narratives.* Suppl 11, s.l. : BioMed Central Ltd, 2008, BMC bioinformatics, Vol. 9, p. S3.

27. **Bustio, L., C., Hernández and René, C.** *FPGAs en el entrenamiento de SVM para clasificación de grandes volúmenes de datos.* s.l. : CENATAV, 2009. Vol. Serie Gris, Reporte ténico Minería de Datos.

28. *Text categorization with support vector machines: Learning with many relevant features.* **Joachims, T.** s.l. : Springer, 1998, Machine Learning: ECML-98, pp. 137-142.

29. *Improvements to Platt's SMO algorithm for SVM classifier design.* **Keerthi, S., et al.** 3, s.l. : MIT Press, 2001, Neural Computation, Vol. 13, pp. 637-649.

30. *Sequential minimal optimization: A fast algorithm for training support vector machines.* **Platt, J.** s.l. : Citiseer, 1998.

31. *A tutorial on support vector regression.* **Smola, A. and Schölkopf, B.** 3, s.l. : Springer, 2004, Statistics and Computing, Vol. 14, pp. 199-222.

32. eXtreme Programming. *Wikipedia.* [Online] ://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema.

33. Scrum. *Wikipedia.* [Online] http://es.wikipedia.org/wiki/Scrum.

34. *Agile development: Lessons learned from the first scrum.* **Sutherland, J.** 20, 2004, Cutter Agile Project Management Advisory Service: Executive Update, Vol. 5, pp. 1-4.

35. Lenguaje de programación Java. *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java.

36. Extensible Markup Language (XML). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Extensible_Markup_Language.

37. Document Object Model (DOM). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Document_Object_Model.

38. XML Path Language (XPath). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/XPath.

39. Lof4j. *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Log4j.

40. Concurrent Versions System (CVS). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/CVS.

41. XML Schema (XSD). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/XML_Schema.

42. Castor. [Online] http://www.castor.org/.

43. OpenKapow. [Online] http://www.openkapow.com/.

44. Representational State Transfer (REST). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Representational_State_Transfer.

45. Apache. *HTTP Components: HTTP Client.* [Online] http://hc.apache.org/httpclient-3.x/.

46. Comma-separated values (CSV). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/CVS.

47. Lucene. *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/Lucene.

48. Snowball. [Online] http://snowball.tartarus.org/index.php.

49. Sourceforge. *Web-Harvest.* [Online] http://web-harvest.sourceforge.net/.

50. XQuery. *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/XQuery.

51. eXtensible Hypertext Markup Language (XHTML). *Wikipedia (The Free Encyclopedia).* [Online] http://es.wikipedia.org/wiki/XHTML.

52. **Wishart, D., et al.** DrugBank: a knowledgebase for drugs, drugs actions and drug targets. *Nucleic acids research.* 2007.

53. *The Unified Medical Lnaguage System.* **Lindberg, D., Humphreys, B. and McGray, A.** 4, 1993, Methods of information in Medicine, Vol. 32, p. 281.

54. *The unified medical language system (UMLS): integrating biomedical terminology.* **Bodenreider, O.** Database Issue, s.l. : Oxford Univ Press, 2004, Nucleic Acids Research, Vol. 32, p. D267.

55. *Medical subject headings (MeSH).* **Lipscomb, C.** 3, s.l. : Medical Library Association, 2000, Bulletin of the Medical Language System, Vol. 88, p. 265.

56. **Spackman, K., Campbell, K. and CÃ, R.** SNOMED RT: a reference terminology for health care. [ed.] American Medical Informatics Association. *Proceedings of the AMIA Annual Fall Symposium.* 1997, p. 640.

57. **Aronson, A.** Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proceedings of the AMIA Symposium.* s.l. : American Medical Informatics Association, 2001, p. 17.

58. *Corpus annotation schemes.* **Leech, G.** 4, s.l. : ALLC, 1993, Literary and linguistic computing, Vol. 8, p. 275.

59. C4.5 Algorithm. *Wikipedia (The Free Encyclopedia).* [Online] http://en.wikipedia.org/wiki/C4.5_algorithm.

60. Naive Bayes Classifier. *Wikipedia (The Free Encyclopedia).* [Online] http://en.wikipedia.org/wiki/Naive_Bayes_classifier.

61. Linked Data. [Online] http://linkeddata.org/.

62. *Expanding the Pathway and Interaction Knowledge in Linked Life Data.* **Momtchev, V., et al.**

ANEXO I

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Víctor Méndez González

2.- Departamento:

Informática

3.- Descripción del Proyecto:

- Titulo	**Ingeniería Técnica en Informática de Gestión**
- Duración (meses)	**18**
Tasa de costes Indirectos:	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Doctor 1		Doctor	1	6.000,00	6.000,00	
Ingeniero 1		Ingeniero Senior	3	4.289,54	12.868,62	
Ingeniero 2		Ingeniero	3	2.694,39	8.083,17	
					0,00	
					0,00	
		Hombres mes	**7**	**Total**	26.951,79	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
		100		60	0,00
		100		60	0,00
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	0,00

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	26.952
Amortización	0
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	5.390
Total	32.342